Optimised Scheduling of Online Experiments

Eugene Kharitonov[1,2], Craig Macdonald[2], Pavel Serdyukov[1], Iadh Ounis[2]
[1]Yandex, Russia
[2]University of Glasgow, UK
[1]{kharitonov, pavser}@yandex-team.ru
[2]{craig.macdonald, iadh.ounis}@glasgow.ac.uk

ABSTRACT

Modern search engines increasingly rely on online evaluation methods such as A/B tests and interleaving. These online evaluation methods make use of interactions by the search engine's users to test various changes in the search engine. However, since the number of the user sessions per unit of time is limited, the number of simultaneously running on-line evaluation experiments is bounded. In an extreme case, it might be impossible to deploy all experiments since they arrive faster than are proccessed. Consequently, it is very important to efficiently use the limited resource of the user's interactions. In this paper, we formulate the novel problem of schedule optimisation for the queue of the online experiments: given a limited number of the user interactions available for experimentation, we want to re-order the queue so that the number of successful experiments is maximised. In order to build a schedule optimisation algorithm, we start by formulating a model of an online experimentation pipeline. Next, we propose to reduce the task of finding the optimal schedule to a learning-to-rank problem, where we require the most promising experiments to be ranked first in the schedule. To evaluate the proposed approach, we perform an evaluation study using two datasets containing 82 interleaving and 35 A/B test experiments, performed by a commercial search engine. We measure the quality of a schedule as the number of successful experiments executed under limited user interactions. Our proposed schedulers obtain improvements of up to 342% compared to the unoptimised baseline schedule on the dataset of interleaving experiments and up to 43% on the dataset of A/B tests.

Categories and Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

Keywords: online evaluation; interleaving; A/B tests

1. INTRODUCTION

Online experimentation methods have increasingly attracted attention from researchers and practitioners. The major search engines such as Google [22] and Bing [12] have developed large-scale online experimentation pipelines. Kohavi et al. [12] claimed that the number of A/B tests deployed by Bing grew exponentially over the years.

We define an online *experiment* as a single unit of the online evaluation that aims to test a single change in the search engine. An online experiment might span weeks and affect millions of user sessions. An experiment where the tested change proved to improve a considered metric with respect to the baseline system is further referred to as a *successful experiment*. We assume that obtaining a large number of such successful experiments per unit of time is important for the evaluation pipeline as a whole, as it allows a search engine to evolve quickly. In a typical scenario, several groups work on improving the search engine's ranking simultaneously and in different areas (e.g. ranking algorithms, ranking features). However, when tested online, the majority of their changes turn out to be useless or harmful [10].

Since the number of experiments grows with the intensity of the search engine development, after some point, these experiments need to "compete" for a limited resource of user interactions available to the search engine. These observations lead us to the idea of optimising the order of the online experiments: we need to order the schedule of the experiments so that the most promising experiments are performed first. Indeed, the earlier a successful comparison is performed, the earlier the corresponding change will be deployed. In an extreme case, when the experiments are arriving faster than they are proccessed, it is also beneficial to schedule only the promising experiments, without spending resources on the less promising ones.

In this paper, we study the problem of the online experiment schedule optimisation. We start with formally describing the model of the experimentation pipeline and the requirements for the scheduling algorithms. Further, we concentrate on the effectiveness-related experiments, where the changes that affect the ranking of the results are tested. We propose to reduce the problem of the optimal scheduling of such experiments to a learning-to-rank problem, considering pointwise and pairwise formulations. We describe a rich feature representation of the online experiments, used in the machine learning step. Finally, we perform a thorough evaluation study of the efficiency of the resulting scheduling algorithms. The contributions of this work are three-fold:

- We formulate the novel problem of the optimal scheduling of online experiments;
- We propose to reduce the problem of the optimal scheduling of the experiments to a learning-to-rank problem, and consider pointwise and pairwise formulations of this learning-to-rank problem;
- We thoroughly evaluate the proposed scheduling algorithms.

The remainder of this paper is organised as follows. After discussing the related work in Section 2, we briefly review how online experiments are typically performed and describe our experimentation pipeline in Section 3. In Section 4 we discuss how the scheduling problem can be reduced to a learning-to-rank machine learning problem. In Section 5 we discuss the datasets we use in our evaluation study. The evaluation methodology is described in Section 6. The evaluation results we obtained are discussed in Section 7. We conclude the paper and discuss future work in Section 8.

2. RELATED WORK

Our work relates to existing research in improving the efficiency of online experimentation, as discussed below.

Tang et al. [22] described the multi-layer framework of overlapping experiments used in Google. The motivation of this framework is to build a scalable online experimentation mechanism. The core idea of the framework is to use different *layers* of experiments, with each layer being associated with a set of variables influencing the user experiments. This allows each user interaction to participate in several experiments simultaneously. For instance, one of these experiments can compare two ranking algorithms, another can evaluate changes in the UI, etc. A similar framework is used by the Bing search engine [12].

Our work shares the same goal: to make the online experimentation pipeline more efficient, but extends [22] further. Even in the framework proposed by Tang et al., in some layers (e.g. the ranking layer) we might want to run the most promising experiments earlier or experiments might arrive faster than they could be processed. As a result, the problem of the optimal scheduling of these experiments is still an issue. We consider a complementary approach: assuming that only a part of the experiments can be successful, we propose to schedule the queue of the experiments so that more successful experiments are performed.

Notable efforts [9, 18, 23] have been applied to improve the sensitivity of the interleaving experiments, so that each interleaving experiment can be finished faster. Yue et al. [23] proposed a machine-learned approach to interpret click feedback from the users, so that the intrinsic noise of the clicks is reduced. Chapelle et al. [1] additionally considered some simple heuristic weighting schemes that can be used while aggregating the user's clicks in a single impression. We study a complimentary approach to increase the experimentation efficiency: under the optimising schedule, more successful experiments can be performed.

An important step in our approach is to predict how likely a particular experiment is to be successful. Hofmann et al. [4] proposed to estimate the interleaving comparison outcomes by treating historical user sessions as comparison events between tested alternatives. Kharitonov et al. [9] used the historical click data to calculate the expected difference in the number of clicks each interleaving alternative receives after an interleaving experiment is deployed. This step can be considered as a predictor of the experiment's result. In a recent work, Li et al. [14] proposed to leverage historical click data and natural variance in the search engine's result pages to predict the results of A/B tests.

In this work, we also use historical click data to predict the interleaving experiment outcome. However, there are considerable differences with the above discussed work [4, 9, 14]. Indeed, predicting an outcome of an experiment is just one of the steps of our proposed experiment scheduling approach. Moreover, the historical click data forms only a part of the features we use in our study: we additionally consider features that are based on the offline effectiveness-based evaluation, and online exploration.

Radlinski and Craswell [17] studied the agreement between the offline evaluation metrics, such as nDCG@n or Precision@n, and the results of interleaving experiments. Their work is related to our research, since they demonstrated that some metrics, such as nDCG@n, have a statistically significant correlation with the outcomes of interleaving experiments. This fact implies that offline evaluation metrics can be useful in predicting the interleaving experiment results. A similar experiment was performed by Chapelle et al. [1], who measured the correlation between DCG@5 and the absolute online metrics used in A/B tests.

However, since the agreements reported in [1, 17] are not perfect, the following question arises: Can a better prediction can be achieved by using other features apart from the search result effectiveness? In our evaluation study we address this question. Further, Radlinski and Craswell [17] performed their study on a dataset containing three experiments with major changes, and two experiments with minor improvements. However, major changes are rare in modern commercial search engines. Similarly, the analysis of the absolute online metrics used in A/B tests, performed by Chapelle et al. used a dataset of 6 comparisons. In contrast, we use two datasets, containing 82 real-life interleaving experiments and 35 A/B tests, performed by a commercial search engine as part of its development. Thus, we argue that our study uses more representative datasets.

Overall, to the best of our knowledge, our paper is the first work that studies the optimisation of the schedule of the online experiments. However, our work finds a solid foundation in the research discussed above.

3. ONLINE EXPERIMENTATION

In the next section we shortly review how online experiments are performed (Section 3.1) and the model of the experimentation pipeline we consider (Section 3.2).

3.1 Evaluation Methods

Suppose, we want to compare two different ranking algorithms, referred to as A (the current system) and B (the tested improvement). There are two popular online evaluation approaches to do that: A/B tests and interleaving.

A/B tests An intuitive way to compare two systems is to split users in two groups, the treatment and the control group. Each group of users is served by one of the algorithms (A or B). After collecting the user behaviour data in the experiment e, an online user satisfaction metric σ is calculated for A ($\sigma_A(e)$) and B ($\sigma_B(e)$). If the collected data suggests that the alternative B demonstrates statistically significant improvement in the selected user metric (i.e. $\sigma_B(e) >_{st.s} \sigma_A(e) \Rightarrow B \succ A$) then the experiment is considered as successful. We use the clickthrough rate online metric, which indicates the ratio of the clicked result pages in the query stream. The inverse of the clickthrough rate, the abandonment rate is frequently used in the literature, e.g. [1, 2, 19]. Our approach does not rely on the specific properties of the metric used, so it can be applied with any of the existing online metrics.

Interleaving In an interleaving experiment, result pages are generated by interleaving (mixing) results from both alternatives. Next, the user's click feedback is interpreted in order to derive which of the tested algorithms provides the users with a better results ranking.

While the scheduling approaches discussed in this paper can be applied to any of the existing interleaving methods [5, 8, 17, 18], in our evaluation study we use a dataset of Team Draft-based [17] interleaving experiments. The Team Draft modification we use leverages the *deduped binary* click aggregation scheme [1], which proved to be sensitive, i.e. allowing a faster convergence rate. In a user interaction, alternative A (B) is assigned with credit equal to the total number of clicks its results received in the interaction. If the top k results in A and B are identical, then the clicks on these results are ignored. After that, in each user interaction the alternative that has more credit is defined as a winner. If both alternatives have equal credits, the result of the comparison in this interaction is a tie. In an experiment e we denote the number of user interactions with ties as $ties(A, B)$, the number of interactions won by A (B) as $wins(A)$ $(wins(B))$. The outcome of the experiment e is the following quantity [1]:

$$s(e) = \frac{wins(B) + \frac{1}{2}ties(A,B)}{wins(A) + wins(B) + ties(A,B)} - \frac{1}{2} \qquad (1)$$

If $s(e)$ is significantly above zero, then it is concluded that B outperforms A ($B \succ A$) and the experiment e is successful.

3.2 Experimentation pipeline

In this paper, we work under a particular case of the experimentation pipeline. We specify this model by describing three assumptions, as discussed below:

A1 *The upper bound of the user interactions available for each experiment (the experiment's budget) is pre-defined and equal to T.*

In practice, usually the part of the query stream used for a single experiment is fixed and set to several percent [12]. At the same time, the size of the query traffic of a search engine is influenced by various factors, including the time of the day, the day of the week, and the season of the year. However, we assume that the experiments are deployed long enough that the per-day variations of traffic are averaged out, while the seasonal variations are smooth enough not to influence the size of the traffic while experiments are performed. This can be achieved by fixing the duration of the experiments to be of a size of a week or two [12]. This assumption allows us to simplify the schedule planning and evaluation steps.

A2 *Once an experiment is started, it is never interrupted.*

After an experiment is started it is never stopped (such an interruption is referred to as preemption [13]) until one of the two outcomes is achieved: one of the alternatives (A or B) wins the comparison, or the experiment's budget n is entirely exhausted. In the case of interleaving experiments, where no *early stopping* scheme is available, we assume that the experiment is always performed until its budget is exhausted. On the other hand, in the case of A/B tests, an experiment might be stopped before using all interactions if an early stopping mechanism is used, such as the scheme proposed in Section 4.3 of [12].

Notably, a currently running experiment is never stopped, even if a new experiment comes to the queue and it turns out to be more promising. Although this restriction can result in a sub-optimal use of the user interactions, this assumption ensures some desired properties of the experiments. First, it is generally accepted to deploy online experiments for an integer number of weeks of continuous time, so that every week day is represented in the experimental data [12]. Thus the situation where a 7 day experiment is started on Monday, stopped on Tuesday, and continued next Monday is not desirable as not each day of the week is covered. Moreover, interrupting an A/B experiment with another experiment might result in increased variance in the observations (due to the carry-over effect [11]), and thus reduce the sensitivity of the experiments. At the same time, most of the changes at search engines are relatively small and hence are difficult to detect immediately. As a result, the metrics used in A/B tests should be measured for a relatively long and continuous period of time. Finally, this assumption makes the whole pipeline predictable and understandable, which is an important property of a production-level system.

As new experiments continually arrive in the queue, it is possible that an old experiment will not be executed for a long time, i.e. it *starves*. The question arises as how to handle this case: should we prioritise old experiments over new ones? In this paper, we work with the following assumption:

A3 *It is acceptable for some experiments to starve. The experiments should "compete equally" no matter how long ago they arrived in the queue.*

This assumption simplifies modelling the queue and makes the scheduling algorithm easy to understand and predict. To alleviate the consequences of infinite starving in a real-life production setting, the scheduling algorithm can be accompanied by a manually handled queue. Experiments that are essential to be deployed despite the predictions of the scheduler can be deployed in this manual queue, if necessary.

The task of the scheduler we study is to sort the queue of the experiments, so that the number of successful experiments performed under the limited number of the user interactions is maximised. When studying this task, the exact implementation of the experimentation pipeline does not play an important role. We only require A1-A3 to hold. For instance, two experiments can run parallel for two weeks, using 5% of the user interaction stream each, or the first experiment can be deployed for a week on 10% of the traffic, followed by the second experiment. From the scheduling point of view, we do not differentiate between these two cases.

4. OPTIMISING THE SCHEDULE

In this section, we firstly formulate the problem of optimised scheduling (Section 4.1), using the assumptions discussed in Section 3.2. Since this formulation relies on feature-based machine learning, in Section 4.2 we describe the feature representation of the online experiments we use. Finally, in Section 4.3 we describe our approach to reduce the scheduling problem to a learning-to-rank problem.

4.1 Scheduling Model

The aim of the scheduler is to maximise the number of successful experiments performed. We firstly define "the probability of success" $P(e)$ for an experiment $e \in \mathbb{E}$. This probability is equal to the frequency of the experiment's success (B winning A), if it was repeatedly deployed. Given a fixed schedule S, the expected number of successful experiments is equal to

$$O = \sum_{i=1}^{|\mathbb{E}|} P(S(i)) \mathbb{I}\left[\sum_{j=1}^{i} n_{S(j)} \leq T\right] \qquad (2)$$

where T is the number of user interactions available for the experimentation, $n_{S(j)}$ is the number of interactions used for the jth experiment in the schedule ($S(j)$), and $\mathbb{I}(\cdot)$ is the indicator function. To optimise the number of successful experiments, we use a greedy, shortest job first-like, scheduling

Input: Set of sessions Q
Output: The click model parameters for each document u: a_u, s_u

$a_u^N \leftarrow 0$; $a_u^D \leftarrow 0$
$s_u^N \leftarrow 0$; $s_u^D \leftarrow 0$
Init beta priors: $\alpha_a = 1, \alpha_s = 1, \beta_a = 1, \beta_s = 1$
foreach *session* $s \in Q$ **do**
  foreach *result* u *above or on the last clicked position* **do**
    $a_u^D \leftarrow a_u^D + 1$
  end
  foreach *clicked result* u **do**
    $a_u^N \leftarrow a_u^N + 1 \quad s_u^D \leftarrow s_u^D + 1$
  end
  $u \leftarrow$ last clicked document in s
  $s_u^N \leftarrow s_u^N + 1$
end
foreach u **do**
  $a_u \leftarrow \frac{a_u^N + \alpha_a}{a_u^D + \alpha_a + \beta_a} \quad s_u \leftarrow \frac{s_u^N + \alpha_s}{s_u^D + \alpha_s + \beta_s}$
end

Algorithm 1: Training the sDBN model, as described by Chapelle et al. [3].

algorithm. This algorithm prioritises the interleaving experiments such that the experiments that are ranked higher are expected to be likely more successful. Overall, the greedy scheduling algorithm can be organised as follows. Suppose, a set of the experiments $\mathbb{E}_0$ is available in the queue, currently ordered according to schedule $S_0 = \{e_{S_0(1)}, ..., e_{S_0(|\mathbb{E}_0|)}\}$. Importantly, the queue is ordered so that if an experiment e_i is ranked earlier in the schedule than another e_j, then this necessarily implies that the first experiment e_i has a higher probability of being successful. Denoting the position of the experiment e in the queue S as $S^{-1}(e)$, this requirement can be formalised as follows:

$$S_0^{-1}(e_i) < S_0^{-1}(e_j) \Rightarrow P(e_i) \geq P(e_j) \tag{3}$$

In the next step, a set of new experiments $\mathbb{E}_{new}$ arrives in the queue: $\mathbb{E}_1 = \mathbb{E}_0 \bigcup \mathbb{E}_{new}$. After that, for each new experiment e, the estimate of its probability of success $P(e)$ is calculated. A new schedule S_1 is obtained by sorting the full set of experiments $\mathbb{E}_1$ so that Equation (3) holds. Once a currently running experiment finishes, the firstly scheduled algorithm ($S(1)$, with the highest value of $P(e)$) is deployed.

Under the assumptions A1-A3 this algorithm is optimal[1]:

FACT 1. *Assume the probability of success $P(e)$ is available for each experiment $e \in \mathbb{E}$. Then the greedy scheduling algorithm is optimal, i.e. it maximises Equation* (2).

To run this greedy algorithm, a procedure to estimate the probability of an experiment's success is required. To build such an algorithm, we propose to use a learning-to-rank approach. Further, we discuss the feature representation used in our work (Section 4.2) and the machine learning algorithms (Section 4.3).

4.2 Features

We divide our features into three groups: effectiveness-based features, click model-based features, and the online exploration features. All of these features characterise a particular pair of compared systems A and B. For this reason, most of the features (except for one, interleaving-specific, feature) can be used for scheduling A/B tests and interleaving experiments.

Effectiveness-based group (12 features) The commonly accepted way to evaluate the difference between two ranking algorithms is to assess their quality within the offline evaluation paradigm. Under this paradigm, a set of previously labelled queries are submitted to both alternatives. After retrieving the search result lists (SERPs), they are intersected with the available document labels. Finally, the quality of the ranking is represented by one of the offline metrics, such as Precision@N, ERR@N [2], DCG@N [6].

To get a fine-grained representation of the experiments, we vary the cut-off depth and the way the unlabelled documents are treated while calculating these metrics. We calculate the average values of the metrics for both alternatives, while considering non-labelled documents as irrelevant. Next, we calculate the averages of the same metrics only for queries where both alternatives (A and B) have all top-N documents labelled. This procedure was applied to Precision@1, Precision@3, ERR@3, DCG@3, ERR@5, and DCG@5 metrics. To get a feature representation for an experiment e from the averages of the metrics, we calculate the differences between the averaged values over the queries of the metrics for both alternatives tested in the experiment (e.g. the difference between the averaged values of Precision@1 of alternatives A and B is a feature). Thus each experiment is associated with (Precision@1, Precision@3, ERR@3, DCG@3, ERR@5, DCG@5) $\times$ (unlabelled documents are treated as irrelevant, or the corresponding pairs of SERPs are ignored) $= 6 \times 2 = 12$ effectiveness-based features.

Click model-based group (3 features) The relevance judges can misinterpret queries and misunderstand the user's intentions. A possible way to address this is to use implicit feedback from the real users. In this work, we use pre-trained user click models to predict how users will behave once they are presented with a result page from A or B. Specifically, we train a Simplified Dynamic Bayesian Network (sDBN) [3] click model using a separate part of the dataset. Under the sDBN click model, for a fixed query, each document u has two parameters defined: the probability of the user clicking on the document if it was examined (attractiveness) a_u, and the probability that the document will satisfy the user, if it was clicked, s_u. These parameters are calculated by Algorithm 1, proposed by Chapelle et al. [3].

After that, we use this pre-trained click model to calculate the following features. First, we calculate the difference in the probabilities of the user satisfaction (as defined by the sDBN model) by the result pages of A and B (Algorithm 2). We calculate the value of this difference for two cut-off levels, considering three and five top-ranked results from both alternatives. We use only the top-ranked results, as they are likely to have sufficient click data in the logs. Second, we calculate the expected difference in the number of clicks for alternatives A and B in the interleaving experiment (Algorithm 3). The latter feature is used only for the scheduling of the interleaving experiments, as it is designed to simulate an interleaving experiment. We use a cut-off level of five.

In Algorithms 2 & 3 we use the following notation. The set of interleaved result lists generated for the query is denoted as $\mathbb{L}$, and the probability of showing the interleaved result list L_i to the users is f_i.

Online exploration group (1 feature) A completely different approach to gain useful information about an experiment e is to perform a preliminary deployment for a short period of time. After this, we calculate a feature representing

[1]Due to space constraints, the proof is omitted here.

Input: Parameters of the click model, a_u, s_u; the result page R, the cut-off level k
Output: The probability of the user's satisfaction with R, P_{sat}
$//P_{dsat}$ is the probability of the user's dissatisfaction
$P_{dsat} \leftarrow 1$
for $r \leftarrow 1$ *to* $min(k, |R|)$ **do**
  // u is the document on the rth position
  $u \leftarrow R(r)$
  //update the probability of dissatisfaction
  $P_{dsat} \leftarrow P_{dsat} \cdot (1 - a_u s_u)$
end
$P_{sat} \leftarrow 1 - P_{dsat}$

Algorithm 2: Calculating the probability of the user's satisfaction with the result page R.

the outcome of this preliminary experiment (Equation (1) for interleaving, and $\sigma_B(e) - \sigma_A(e)$ for A/B tests). To calculate this feature in our model, we sample a pre-defined number of user interactions from the experiment data. As the number of interactions sampled can greatly influence the prediction quality (i.e. if we sample sufficiently enough interactions while doing exploration we might not need the experiment itself), in our empirical study we vary this number to gain additional insights into the relative usefulness of this feature. This exploration step is akin to the exploration step in the muti-armed bandit setup [21]. Indeed, given a set of experiments ("arms"), we need to identify which of them is more likely to be successful. However, the pipeline requirements we work under do not permit the use of the existing bandit algorithms, as A2 assumes that a deployed experiment cannot be interrupted.

Feature aggregation After calculating the effectiveness-based and click model-based features, we additionally aggregate them by averaging over four groups according to the query length measured by the number of space-separated terms: (1) all queries, (2) queries of length of 1, (3) queries of length of 2, (4) queries of length of 3 and longer. The exploration feature is not included in this aggregation step. Our intuition behind this feature aggregation step is that it allows the machine-learned algorithms to detect cohorts of queries where the main change in the experiment occurs. For instance, if major relevance changes are observed in the group of long queries, which are likely to be rare, the click model-based features can be less useful. As a result of this aggregation, each interleaving experiment $e \in \mathbb{E}$ is represented as a point in a space of $(12+3) \cdot 4 + 1 = 61$ features. Similarly, each A/B test is represented as a point in a feature space with $(12+2) \cdot 4 + 1 = 57$ dimensions.

Different sets of features might be useful in different scenarios. For instance, in some interleaving experiments, such as those that test changes in the personalisation algorithms, the personalised relevance labels might be unavailable. In contrast, the exploration-based feature can be valuable in this case. On the other hand, the click model-based features can be less useful for the experiments where only the ranking of the long-tail queries is affected. We argue that combining all these groups of features can improve the performance.

4.3 Learning Framework

To apply a greedy scheduling algorithm, we need to estimate the experiments' probability of success. We consider this problem as a ranking problem, and further discuss pointwise and pairwise learning-to-rank approaches to it.

Table 1: Descriptive statistics of the datasets.

Dataset	#Exp	B wins A	Impressions: Min	Median	Mean	Max	Total
Interleaving	82	31	178K	1M	2M	39M	174M
A/B tests	35	10	1.4M	42M	54M	196M	1899M

Input: Parameters of the click model, a_u, s_u; the set of interleaved result lists, $\mathbb{L}$, the cut-off level k.
Output: The expected difference D in the number of clicks obtained by alternatives A and B.
$//P_e(r)$ denotes the probability of examining the rth position
$P_e(1) \leftarrow 1$
foreach $L_i \in \mathbb{L}$ **do**
  for $r \leftarrow 1$ *to* $min(k, |L_i|)$ **do**
    // u is the document on the rth position
    $u \leftarrow L_i(r)$
    //update the expected difference
    $D \leftarrow D + f_i P_e(r) a_u (\mathbb{I}[r \ from \ A] - \mathbb{I}[r \ from \ B])$
    //probability of examining the next document
    $P_e(r+1) \leftarrow P_e(r)\,(1 - a_u s_u)$
  end
end

Algorithm 3: Calculating the expected difference in clicks obtained by A and B in an interleaving comparison.

Pointwise A simple approach to predict the experiment's probability of being successful is to train a classifier that discriminates successful experiments from others. We associate experiments with one of the two classes $\{0, 1\}$: $y(e) = \mathbb{I}[B \succ A]$. Informally, the experiments for which B statistically significantly outperforms A are considered as instances of the positive class 1. All other experiments, including those where no statistically significant difference between A and B was found, belong to the negative class 0. In the second step, we train a machine learning algorithm to predict the class of the considered experiment. We use two popular methods to build such a binary classifier. First, we use logistic regression with L1 regularisation [16]. The regularisation parameter is tuned by a ten-fold cross-validation on the training set. Second, we use the gradient boosted trees algorithm provided in the GBM package for R [20]. The parameters (e.g. number of trees) are tuned throught cross-validation.

Pairwise In the pointwise approach all positive examples are treated equally. However, in some of the experiments, the difference between the alternatives is bigger. With everything else being equal, it is better to deploy such experiments earlier, as the corresponding search engine's improvement is larger. This idea is naturally represented by the pairwise learning-to-rank paradigm. We firstly define a set $\mathbb{P}$ of pairs of experiments $(e_i, e_j) \in \mathbb{E} \times \mathbb{E}$ such that the outcome of the first experiment in the pair is higher than that of the outcome of the second. $\mathbb{P}$ formulated as follows:

$$\mathbb{P} = \{(e_i, e_j) : \sigma(e_i) > \sigma(e_j), e_i, e_j \in \mathbb{E}\}$$

where $\sigma(e) = s(e)$ for the interleaving experiments, and $\sigma(e) = \sigma_B(e) - \sigma_A(e)$ in the case of A/B tests.

We use the GBM [20] package for R to find the function $f(r_e)$ that minimises the number of misordered pairs. As an alternative pairwise ranker we use RankingSVM [7].

5. DATASETS

Before discussing the experimental study in the next section, we briefly describe the datasets used in this paper.

The first dataset consists of the subset of the interleaving experiments performed by Yandex during a five week period in Spring 2014. The sample contains 82 interleaving experiments, 31 of which were successful (the alternative B outperformed A statistically significantly, $p < 0.05$). The experiments test changes in the non-personalised ranking of the search engine. The second dataset contains 35 A/B tests, performed over a period of two months in Winter/Spring 2014. In 10 A/B tests, B outperformed A by a statistically significant margin ($p \leq 0.05$). On average, the A/B tests and interleaving experiments were deployed for 10 days.

Salient statistics of the datasets are provided in Table 1. The number of interactions per experiment varies, as each experiment was selected to be deployed for time periods of different length, or for different shares of the query stream. The interleaving experiments were performed using the Team Draft interleaving method with the deduped binary scoring scheme, as described in Section 3. The parameters of the sDBN model [3] used for calculating the click model-based features (Section 4.2) are estimated using a separate query log sample from a two-week period. All experiments in the datasets were deployed after this period. All online experiments were deployed on the Russian market.

6. EVALUATION METHODOLOGY

Since our proposed scheduling algorithm relies on predicting the outcomes of the online experiments, we split our evaluation study in two steps: (1) evaluating the experiment outcome prediction, (2) evaluating the quality of the schedules obtained by our proposed scheduling algorithms. We formulate three research questions that we aim to address:

(**RQ1**) *Is it possible to predict the outcomes of the online experiments using the pre-experimental data only, so that the quality of the predictions is improved in comparison with the random order?*

(**RQ2**) *What are the best performing prediction algorithms? How do the proposed features compare to each other?*

(**RQ3**) *How do the learned approaches compare in terms of the quality of the schedules they generate?*

As we discuss in Section 6.4, the random order we compare to in **RQ1** is not an artificial baseline, instead it reflects the stochastic order of the experiments arriving to the experimentation pipeline if no scheduling is performed. In other words, we consider the performance of the randomised order to be similar to the performance of an unoptimised schedule. Notably, each of the groups of features, discussed in Section 5.2, can be considered as a simple predictor of the experiment outcome. Indeed, the experiments in the queue can be ordered according to their DCG scores, or the experiment outcome prediction, based on the click modelling, or based on the results of the exploration step. To obtain additional insights into the relative importance of the features, we additionally investigate the performance of the schedulers that sort experiments according to the separate features.

6.1 Prediction quality

Since our goal is to schedule the successful experiments with a higher priority, it is natural to measure the quality of the schedule ranking as the fraction of the correctly ordered pairs of the experiments. In an ideal ranking, successful experiments ($B \succ A$) are deployed first. Moreover, it is natural to require the successful experiments with higher difference between A and B to be scheduled earlier. This idea can be represented by the Area Under Curve (AUC) quality metric of a classifier S separating successful experiments from unsuccessful ones [15].

Input: Number of user interactions available for an experiment n; the total number of available user interactions T; A set of experiments $\mathbb{E}$
Output: Estimated values of $AUC(S)$ and $Q(S)$.
$A \leftarrow 0$; $Q \leftarrow 0$
foreach $train, test \leftarrow$ $RandomStratifiedSplit(\mathbb{E}, nSplits = 100)$ **do**
  Train the ranker: $C \leftarrow fit(train)$
  Greedily schedule the test experiments: $S \leftarrow predict(C, test)$
  Update the AUC estimate: $A \leftarrow A + AUC(S)$
  Initialise the estimate of Q for the current split: $Q_i \leftarrow 0$
  Calculate the bootstrapping estimate Q_i:
  while $i < N$ **do**
    The remaining experimentation budget: $\hat{T} \leftarrow T$
    Starting with the first experiment: $j \leftarrow 1$
    while $j < |S|$ *and* $\hat{T} > 0$ **do**
      Sample n sessions from experiment $S(j)$
      $data \leftarrow sample(S(j))$
      Check if the experiment is successful, based on $data$:
      if $B \succ A$ **then**
        $Q_i \leftarrow Q_i + 1$
      end
      Reduce the budget by the number of sampled sessions and proceed to the next experiment: $\hat{T} = \hat{T} - n$; $j \leftarrow j + 1$
    end
  end
  $Q \leftarrow Q + \frac{1}{N} Q_i$
end
Calculate averages of the metrics across the splits: $Q(S) \leftarrow \frac{1}{nSplits} Q$, $AUC(S) \leftarrow \frac{1}{nSplits} A$

Algorithm 4: The bootstrap-based evaluation protocol.

We firstly define the set of pairs of experiments $\mathbb{R} = \{(e_{i,1}, e_{i,2})\}_i$ that are used in the evaluation. This set contains all the pairs of experiments such that at least one of the experiments has the alternative B winning the comparison with $p \leq 0.05$ and the relative scores of B are different when compared across experiments. We impose the first requirement as we are not interested in evaluating how good a particular scheduling algorithm is at ranking unsuccessful experiments. The AUC metric $AUC(S)$ of a schedule S can be calculated using the following expression:

$$AUC(S) = \frac{\sum\limits_{e_1, e_2 \in \mathbb{R}} \mathbb{I}\left[\left(S^{-1}(e_1) - S^{-1}(e_2)\right)\left(\sigma(e_1) - \sigma(e_2)\right) < 0\right]}{|\mathbb{R}|} \tag{4}$$

where $\sigma(e) = \sigma_B(e) - \sigma_A(e)$ in the case of the A/B tests and $\sigma(e) = s(e)$ for the interleaving experiments.

6.2 Evaluating the Schedule

While the AUC measure as defined in the previous section is intuitive, it evaluates the quality of the predictions only, not the quality of the resulting schedule. However, there is a noteworthy difference. Indeed, the AUC metric reflects a scenario where there are enough resources (user impressions)

to deploy all the required experiments, but an approach to deploy the promising experiments first is needed. Indeed, a scenario when one cannot deploy all the available experiments due to restricted resources is possible. In this case, AUC is less suitable, as it also measures the quality of the ranking of the experiments that cannot be deployed. Thus, we propose to measure the quality of the scheduling algorithm as the number of the successful experiments it can fit in the number of available user interactions.

Consider a schedule S, representing the order of the experiments to be run, $S = \{e_{S(1)}, e_{S(2)}..., e_{S(|\mathbb{E}|)}\}$. Ideally, the schedule should allow us to run and finish as many experiments where B wins, as possible. At the same time, we have a limited number of the user interactions that can be used in the experiments, denoted T. Thus, we are interested in minimising the following measure, similar to Equation (2):

$$Q(S) = \sum_{i=1..|S|} \left(\mathbb{I}[B(e) \succ A(e)] \cdot \mathbb{I}\left[\sum_{j=1}^{i} n(e_i) \leq T \right] \right) \quad (5)$$

$Q(S)$ measures the number of experiments with B winning the comparison ($\mathbb{I}[B(e) \succ A(e)]$), under the limited number of user interactions T, as only the experiments that are performed before T is reached ($\sum_{j=1}^{i} n(e_i) \leq T$) can contribute to $Q(S)$. Notably, the metric $Q(S)$ can be considered as a generalisation of *Precision@R*. Indeed, if the number of interactions available for each experiment $n(e_i) = n = const$ is large enough for any experiment to have a definite outcome, then $Q(S) \approx precision@R$, where $R = \frac{T}{n}$.

6.3 Statistical Methodology

To evaluate a quality of a schedule S, we perform a bootstrap estimation of the metric values $AUC(S)$ and $Q(S)$ using the datasets described in the previous section. This estimation is performed in several steps. First, we select the experiment that S schedules to run first, $S(1)$. After that, we compare the number of the interactions required to perform the experiment n to the available limit T. If the required number is less than T, we continue, and stop otherwise. From the available dataset of user interactions for the experiment $S(1)$, we perform a bootstrap sampling of the user interactions until the stopping criterion is met. After that, we proceed to the next scheduled experiment, $S(2)$. Again, we sample the user interactions until the stopping criterion is met. We proceed, until the limit T is reached, and calculate the value of $Q(S)$ according to Equation (5). We repeat this described procedure N times and, as a result, it provides us with the bootstrapped estimates of the performance of the tested scheduler.

Further, as the machine learning-based approaches require separated testing and training set, we repeat the described quality estimation algorithm with the train-test split varied. Each split is obtained by randomly selecting 10% of the experiments in the dataset as a test set, with the remaining experiments used for training. The splits are performed in a stratified manner, so that the distribution of the successful experiments is the same in training and test sets. The evaluation algorithm is formally described in Algorithm 4.

We ensure that the total number of user impressions used in the evaluation is equal for all evaluated schedulers. If a scheduler performs an exploration step, we subtract the number of sessions used for exploration from the overall experimentation budget T.

Table 2: Performance of the scheduling algorithms, measured by AUC on the dataset of interleaving experiments. The values in bold outperform other values in the same row, $p < 0.05$ (excluding *UB*).

#sample					
	Rnd	CM	DCG	ERR	UB
–	$0.51_{\pm 0.03}$	$\mathbf{0.77}_{\pm 0.02}$	$\mathbf{0.77}_{\pm 0.02}$	$0.74_{\pm 0.02}$	$1.0_{\pm 0.0}$
	Explore	LR	SVM	LGBM	PGBM
0	–	$0.76_{\pm 0.02}$	$0.72_{\pm 0.03}$	$\mathbf{0.83}_{\pm 0.01}$	$0.81_{\pm 0.02}$
$0.01 \cdot T$	$0.62_{\pm 0.02}$	$0.76_{\pm 0.02}$	$0.73_{\pm 0.02}$	$\mathbf{0.83}_{\pm 0.01}$	$0.81_{\pm 0.02}$
$0.02 \cdot T$	$0.70_{\pm 0.02}$	$0.77_{\pm 0.02}$	$0.74_{\pm 0.02}$	$\mathbf{0.83}_{\pm 0.01}$	$\mathbf{0.82}_{\pm 0.02}$
$0.05 \cdot T$	$0.77_{\pm 0.02}$	$0.77_{\pm 0.02}$	$0.74_{\pm 0.02}$	$\mathbf{0.83}_{\pm 0.01}$	$\mathbf{0.82}_{\pm 0.02}$
$0.10 \cdot T$	$\mathbf{0.83}_{\pm 0.02}$	$0.78_{\pm 0.02}$	$0.75_{\pm 0.03}$	$\mathbf{0.85}_{\pm 0.01}$	$\mathbf{0.84}_{\pm 0.02}$
$0.20 \cdot T$	$\mathbf{0.88}_{\pm 0.03}$	$0.79_{\pm 0.02}$	$0.77_{\pm 0.03}$	$\mathbf{0.86}_{\pm 0.01}$	$\mathbf{0.86}_{\pm 0.02}$

Table 3: Performance of the scheduling algorithms, measured by AUC on the dataset of A/B tests. The value in bold outperforms other values with the same exploration step size (excluding *UB*), $p < 0.05$.

#sample		#sample			
	Rnd	DCG		ERR	UB
–	$0.49_{\pm 0.04}$	$\mathbf{0.56}_{\pm 0.04}$	–	$\mathbf{0.59}_{\pm 0.03}$	$1.00_{\pm 0.00}$
	Explore	LR		Explore	LR
0	–	$\mathbf{0.68}_{\pm 0.04}$	$0.05 \cdot T$	$0.59_{\pm 0.04}$	$\mathbf{0.66}_{\pm 0.03}$
$0.01 \cdot T$	$0.56_{\pm 0.03}$	$\mathbf{0.67}_{\pm 0.04}$	$0.10 \cdot T$	$0.64_{\pm 0.04}$	$0.66_{\pm 0.03}$
$0.02 \cdot T$	$0.54_{\pm 0.03}$	$\mathbf{0.67}_{\pm 0.04}$	$0.20 \cdot T$	$\mathbf{0.73}_{\pm 0.04}$	$0.65_{\pm 0.03}$

6.4 Baselines

Random The first baseline we consider assigns a random order to the online experiments. Since experiments arrive stochastically to the queue, we believe that the performance of this baseline is a good indicator of the average performance of the un-prioritised schedule that never re-arranges the incoming experiments. In other words, this baseline is not an artificial one, but instead it reflects the performance of the real-world schedules. As this paper is the first to address the problem of the online experiment scheduling optimisation, more elaborate baselines do not exist.

UpperBound The *UpperBound* scheduler provides us with an upper bound on the possible scheduler performance. In case of the interleaving experiments, *UpperBound* sorts the experiments in the queue according to the values $s(e)$ of the experiment's outcome (Equation (1)). For the A/B tests data, *UpperBound* sorts the experiments according to the obtained value of $\sigma_B(e) - \sigma_A(e)$. This baseline uses the data produced by the experiment, which is unavailable before the experiment was performed.

7. RESULTS AND DISCUSSION

We use the following notation in this section. The *UpperBound*, and *Random* baseline schedulers are denoted as *UB* and *Rnd*, respectively. The schedulers based on the pointwise predictors, logistic regression and GBM, are referred to as *LR*, and *LGBM*, respectively. The schedulers based on pairwise RankingSVM and GBM, are denoted as *SVM* and *PGBM*, respectively. Finally, the schedulers based on the single DCG@3, ERR@3, and Explore features are referred to as *DCG*, *ERR*, and *Explore*. When calculating the effectiveness metrics used in the effectiveness-based schedulers *DCG* and *ERR*, we consider unlabelled results as non-relevant.

CM denotes the scheduler that ranks experiments according to the click model-based feature, which calculates the expected difference in the number of clicks A and B will obtain in the interleaving experiment. We include it as it

Table 4: The quality of the scheduling algorithms measured by $Q(S)$ on the dataset of the interleaving experiments. The values in bold outperform other in the same scenario (T, #sample), $p < 0.05$ (except for *UB*).

#sample	$T = 3 \cdot 10^5$					$T = 4.5 \cdot 10^5$				
	Rnd	*CM*	*DCG*	*ERR*	*UB*	*Rnd*	*CM*	*DCG*	*ERR*	*UB*
–	$0.60_{\pm 0.05}$	$\mathbf{1.99}_{\pm 0.12}$	$1.04_{\pm 0.13}$	$1.20_{\pm 0.12}$	$2.42_{\pm 0.11}$	$0.58_{\pm 0.06}$	$\mathbf{2.20}_{\pm 0.14}$	$1.37_{\pm 0.11}$	$1.42_{\pm 0.13}$	$2.70_{\pm 0.11}$
	Explore	*LR*	*SVM*	*LGBM*	*PGBM*	*Explore*	*LR*	*SVM*	*LGBM*	*PGBM*
0	–	$1.14_{\pm 0.14}$	$1.45_{\pm 0.14}$	$1.93_{\pm 0.10}$	$\mathbf{2.17}_{\pm 0.08}$	–	$1.39_{\pm 0.16}$	$1.71_{\pm 0.14}$	$2.28_{\pm 0.14}$	$\mathbf{2.56}_{\pm 0.12}$
$0.01 \cdot T$	$0.78_{\pm 0.07}$	$1.14_{\pm 0.13}$	$1.44_{\pm 0.15}$	$1.87_{\pm 0.11}$	$\mathbf{2.18}_{\pm 0.08}$	$0.85_{\pm 0.07}$	$1.37_{\pm 0.13}$	$1.73_{\pm 0.13}$	$2.35_{\pm 0.11}$	$\mathbf{2.54}_{\pm 0.11}$
$0.02 \cdot T$	$0.89_{\pm 0.12}$	$1.13_{\pm 0.10}$	$1.44_{\pm 0.14}$	$1.90_{\pm 0.10}$	$\mathbf{2.13}_{\pm 0.08}$	$1.00_{\pm 0.08}$	$1.39_{\pm 0.13}$	$1.69_{\pm 0.11}$	$2.32_{\pm 0.14}$	$\mathbf{2.54}_{\pm 0.14}$
$0.05 \cdot T$	$1.06_{\pm 0.08}$	$1.12_{\pm 0.10}$	$1.34_{\pm 0.13}$	$1.91_{\pm 0.11}$	$\mathbf{2.10}_{\pm 0.08}$	$1.26_{\pm 0.10}$	$1.44_{\pm 0.14}$	$1.69_{\pm 0.11}$	$2.30_{\pm 0.11}$	$\mathbf{2.50}_{\pm 0.11}$
$0.10 \cdot T$	$1.18_{\pm 0.09}$	$1.04_{\pm 0.10}$	$1.35_{\pm 0.13}$	$1.81_{\pm 0.11}$	$\mathbf{2.02}_{\pm 0.09}$	$1.45_{\pm 0.11}$	$1.36_{\pm 0.14}$	$1.87_{\pm 0.12}$	$2.30_{\pm 0.11}$	$\mathbf{2.50}_{\pm 0.14}$
$0.20 \cdot T$	$1.31_{\pm 0.10}$	$0.94_{\pm 0.11}$	$1.20_{\pm 0.11}$	$1.57_{\pm 0.09}$	$\mathbf{1.83}_{\pm 0.08}$	$1.68_{\pm 0.13}$	$1.26_{\pm 0.08}$	$1.71_{\pm 0.10}$	$2.19_{\pm 0.14}$	$\mathbf{2.46}_{\pm 0.12}$

Table 5: The quality of the scheduling algorithms measured by $Q(S)$ on the dataset of A/B tests. The values in bold outperform other with the same exploration step size (except for *UB*), $p < 0.05$.

#sample			#sample		
	Rnd	UB		DCG	ERR
–	$0.42_{\pm 0.03}$	$0.92_{\pm 0.06}$	–	$0.54_{\pm 0.03}$	$\mathbf{0.61}_{\pm 0.03}$
	Explore	LR		Explore	LR
0	–	$\mathbf{0.60}_{\pm 0.04}$	$0.05 \cdot T$	$0.52_{\pm 0.03}$	$\mathbf{0.57}_{\pm 0.04}$
$0.01 \cdot T$	$0.46_{\pm 0.03}$	$\mathbf{0.59}_{\pm 0.04}$	$0.10 \cdot T$	$0.56_{\pm 0.04}$	$0.56_{\pm 0.04}$
$0.02 \cdot T$	$0.48_{\pm 0.03}$	$\mathbf{0.57}_{\pm 0.06}$	$0.20 \cdot T$	$\mathbf{0.61}_{\pm 0.04}$	$0.57_{\pm 0.04}$

resembles the interleaving experiment outcome prediction feature used in [9] and thus it is interesting to compare to it. However, as it simulates an interleaving comparison, we do not include it as a scheduler for the A/B test evaluation.

As the A/B tests dataset is smaller than the interleaving dataset, and it has considerably fewer successful experiments and thus less training data (Section 5), we found that the use of more elaborated pairwise and GBRT-based techniques did not result in any improvements over a basic logistic regression-based scheduler *LR*. For this reason, in the experiments based on the A/B tests, we report only *LR*.

We use the Wilcoxon test to compare the performance of the schedulers (excluding *UpperBound*). The reported confidence intervals correspond to 95% confidence.

7.1 Prediction Quality

In Tables 2 and 3, we report the results of our evaluation of the quality of the experiment outcome prediction algorithms, measured on the datasets of the interleaving and A/B test experiments, respectively. The quality is measured by AUC, and the measurement is performed by Algorithm 4. The values in bold statistically significantly outperform other values in the same row/exploration step size ($p < 0.05$). We set the number of user interactions T available for running all experiments such that each experiment in the test parts of the datasets is deployed for 10^5 and $2 \cdot 10^6$ interactions for the interleaving experiments and A/B tests, respectively (A/B tests require more user interactions [1] due to lower sensitivity). The number of user interactions used in the exploration step is varied, we report it in the corresponding cells (#sample). We measure the size of exploration step as a fraction of the total number of available impressions T. For instance, if #sample $= 0.05 \cdot T$, then a scheduler uses $0.05 \cdot T$ interactions for exploration. These interactions are uniformly divided among the test experiments. For fairness, we ensure that the same number of impressions is used for each schedule evaluation step, whether the evaluated scheduler uses exploration or not.

From the top parts of Tables 2 & 3, we firstly notice that the baselines demonstrate their expected behaviour. Indeed, the *UB* scheduler achieves the highest performance possible (1.0), and *Rnd* has an AUC close to 0.5, indicating that under a random permutation, the probability of the correct ordering of the pair of experiments is equal to the probability of the inverse ordering. Next, we notice that the effectiveness-based experiment outcome predictors, *ERR* and *DCG*, as well as the click model-based *CM* perform better than the randomised baseline. However, the AUC scores of *DCG*, *ERR*, and *CM* schedulers (e.g. 0.77, 0.74, and 0.77, respectively, Table 2) are far from 1. This implies that there is a considerable room for improvement. Indeed, on examination of the performance of the machine-learned schedulers that do not perform exploration (bottom parts of Tables 2 & 3, #sample = 0), we observe that these schedulers (e.g. *LR* 0.76, *LGBM* 0.83, and *PGBM* 0.81 for the interleaving dataset; *LR* 0.68 for the A/B tests dataset) can achieve a performance higher than that of the *Random* scheduler and of the schedulers based on individual non-exploratory features (*DCG*, *ERR*, and *CM*). On the interleaving experiments dataset (Table 2), the AUC score of the *LGBM* scheduler is 8% better than that of the best of the schedulers that are based on a single feature (*DCG*, 0.83 vs. 0.77, $p < 0.01$), and considerably better than that of the *Random* scheduler (0.83 vs. 0.51, $p < 0.01$). Similarly, on the A/B tests dataset (Table 2), the machine-learned scheduler *LR* outperforms the best effectiveness-based scheduler *ERR* (0.68 vs. 0.59, 15% improvement, $p < 0.01$), and the *Random* scheduler (0.68 vs. 0.49, 63% improvement, $p < 0.01$).

These observations allows us to answer **RQ1**: it is possible to outperform the random scheduler in the task of predicting the experiment outcome by combining different features in the machine learned schedulers, such as *LGBM* and *LR*.

As the number of the user interactions available for exploration grows, the performance of the exploration-based scheduler *Explore* too: on the interleaving experiments dataset, its AUC score starts from 0.62 when $0.01 \cdot T$ interactions are used, and increases up to 0.88 when $0.20 \cdot T$ interactions are used. A similar behaviour is observed on the A/B tests dataset: the AUC score grows from 0.56 to 0.73.

Interestingly, the machine-learned schedulers demonstrate a comparable performance when little or no exploration is

used. Indeed, on the interleaving experiments dataset, *LGBM* achieves an AUC of 0.83 when exploration is not used, which is comparable to the score of *Explore* using $0.10 \cdot T$ interactions for exploration (Table 2, 0.83). Similarly, on the A/B tests dataset, Table 3, *LR* with no exploration obtained an AUC of 0.68, close to the AUC score of *Explore* with $0.1 \cdot T$ interactions used for exploration. This indicates that the use of the machine-learned algorithms can considerably reduce the number of user interactions used in the exploration step.

On comparing the machine-learned schedulers on the interleaving dataset, we note that more advanced *LGBM* & *PGBM* schedulers achieve higher scores than *LR* and *SVM*. This relation holds for all sizes of the exploration step (#sample) we consider.

These observations allow us to answer **RQ2**. On the interleaving experiment dataset, *LGBM* outperforms other schedulers, and it improves over the best effectiveness-based scheduler *DCG* by a margin of 8%. On the A/B tests dataset, among the non-exploratory schedulers, *LR* demonstrates the highest performance with 15% improvement over the best effectiveness-based *ERR* scheduler. When all schedulers are considered on the dataset of A/B tests, *Explore* (#sample $= 0.2 \cdot T$) achieves the highest performance.

7.2 Evaluating the Schedule

In Table 4, we report our results obtained when evaluating the quality $Q(S)$ of the scheduling algorithms on the interleaving experiments dataset. To get additional insights into the performance of the scheduling algorithms, we vary the total number of user interactions available for the experimentation T, $T \in \{3 \cdot 10^5, 4.5 \cdot 10^5\}$. In both cases, after the scheduler ranks the experiments in the test set, we deploy the three[2] experiments ranked first for evaluation. The user interactions left after performing the exploration step are split for these three experiments uniformly, e.g. the non-exploratory schedulers, such as *ERR*, run these three most promising experiments for 10^5 and $1.5 \cdot 10^5$ interactions. The interactions used for exploration are distributed among the test experiments uniformly.

First, we note that the performance of the *Random* scheduler is considerably lower than the upper bound (e.g. 0.60 vs. 2.42, $T = 3 \cdot 10^5$). This indicates that the quality of a random, un-scheduled queue can be markedly improved. Next, we observe that the effectiveness-based schedulers, *DCG* and *ERR*, outperform the random baseline by a considerable margin (e.g. *ERR* 1.42 vs. 0.58, 144% improvement, $p < 0.01$, $T = 4.5 \cdot 10^5$). Consequently, one can get an improved scheduling quality even by simply sorting the queue of the experiments by the effectiveness scores obtained in the offline evaluation. *DCG* performs slightly worse than *ERR* (e.g. 1.37 vs. 1.42, $T = 4.5 \cdot 10^5$). Interestingly, *CM* demonstrates the highest performance among the schedulers that do not use exploration and machine learning, and outperforms *ERR* by a considerable margin (e.g. 1.99 vs. 1.20, $T = 3 \cdot 10^5$). Notably, by performing a short exploration step where only $0.01 \cdot T$ interactions are used, up to 47% improvement might be obtained in comparison with the unscheduled (*Random*) baseline (*Explore* 0.85 vs. *Rnd* 0.58, $T = 4.5 \cdot 10^5, p < 0.01$). This can be extremely useful in the cases where no relevance judgements are available, such as for the evaluation of the personalised ranking algorithms.

[2]We believe this is a reasonable choice, as in each cross validation split the test set contains less than 10 experiments.

The effectiveness-based schedulers are markedly outperformed by the machine-learned scheduling algorithms. The best-performing algorithm, *PGBM*, outperforms the best effectiveness-based scheduler *ERR*, by 81% (2.17 vs. 1.20, $T = 3 \cdot 10^5$, no exploration) and 80% (2.56 vs. 1.42, $T = 4.5 \cdot 10^5$, no exploration). Moreover, *PGBM* outperforms *Random* by 262% ($T = 3 \cdot 10^5$) and 342% ($T = 4.5 \cdot 10^5$).

The quality of the schedule generated by the *Explore* scheduler grows as the number of the user interactions used for exploration grows (0.78, 0.89, 1.06, 1.18, 1.31) for the exploration sample sizes of $T \cdot \{0.01, 0.02, 0.05, 0.10, 0.20\}$, left part of Table 4). The score of 1.31 corresponds to an improvement of 144% with respect to *Rnd*. However, when the machine-learned schedulers are considered (e.g., *PGBM*) the increased exploration actually harms the performance of the scheduler, as interactions are spent on exploration, instead of being spent on running the experiments. The same effect is observed on the right part of Table 4.

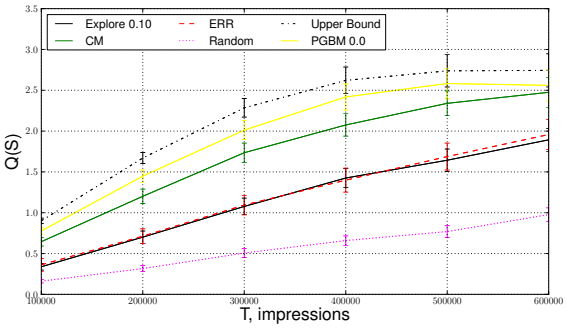
On comparing the results with T varied (left and right parts of Table 4, T is the total number of user interactions available for experimentation) we notice that as T increases, the values of the $Q(S)$ metric increase for all tested schedulers. In particular, the upper bound of the scheduling performance grows from 2.42 to 2.70. This result is intuitive: as we fixed the number of experiments we are attempting to deploy, more user interactions are available for running an experiment, and thus the number of experiments where no statistically significant difference between A and B is detected, is reduced. Some of the experiments "become" successful, and the metric values grow.

In Table 5 we report the $Q(S)$ metric, measured on the dataset of A/B tests. We set the number of interactions available for the experimentation pipeline T to be equal to $4 \cdot 10^6$, and deploy the two first ranked experiments. Similarly to the study performed on the interleaving dataset, we observe that the "natural" stochastic scheduler *Rnd* has a performance considerably lower than the upper bound *UB* (0.42 vs. 0.92). This indicates a possibility of improvement. Indeed, the effectiveness-based schedulers such as *DCG* and *ERR* outperform *Rnd* (e.g., *Rnd* 0.42 vs. *ERR* 0.61). On this dataset, *Explore* with #sample $= 0.20 \cdot T$, *ERR*, and *LR* demonstrate similar performance, and outperform *Random* by 43%. A possible explanation for *LR* performing close to the individual features is the small size of the dataset.

To obtain an additional insight into the behaviour of the schedulers, we vary the overall number of user interactions available to the experimentation pipeline (T) in a pre-defined set $T = m \cdot 10^5, m \in \{1, ..., 6\}$, and measure the quality $Q(S)$ of the resulting schedules. At each step, we try to deploy m experiments scheduled first for evaluation. We run this study only on the interleaving dataset, as it has more experiments. We report the results in Figure 1. From Figure 1 we observe that in all situations *Rnd* is dominated by other schedulers, including the effectiveness-based scheduler *ERR*. *Explore* with #sample $= 0.10 \cdot T$ demonstrates a performance similar to *ERR*. Interestingly, *CM* outperforms both *ERR* and *Explore* by a considerable margin, and is close to *PGBM* that uses exploration. In each situation *PGBM* with no exploration demonstrates the best performance.

We now can answer **RQ3**. On the interleaving experiments dataset, the machine-learned schedulers demonstrate the best performance: *LGBM* outperforms the random baseline by 342% maximum, and the closest effectiveness-based scheduler, *ERR*, by up to 81%. In turn, by using the *ERR*

Figure 1: Quality $Q(S)$ of the best schedulers as the number of the user sessions available grows, measured on the interleaving dataset. *PGBM 0.0* and *PGBM 0.10* are almost indistinguishable.



scheduler, an improvement of 144% over the un-scheduled queue can be obtained. Finally, by performing an exploration step using $0.20 \cdot T$ user interactions, *Explore* improves over the baseline by up to 190% on the interleaving dataset. When the A/B tests dataset is considered, an improvement over the baseline of 45% can be achieved by using the non-exploratory *LR*. A similar improvement can be obtained by using *Explore* that samples $0.20 \cdot T$ interactions.

Our evaluation study allowed us to answer all of the stated research questions. Our obtained results suggest that the random ("natural") order of the experiment schedule can be improved by the greedy scheduling algorithm, both for interleaving and A/B tests. A considerable improvement can be achieved when the greedy scheduling algorithm uses effectiveness measurements (*ERR* and *DCG*), or the pre-experimental click data (*CM*). A further improvement is observed on the interleaving dataset when the greedy scheduling algorithm was used with the predictions from *PGBM*. Finally, *Explore* demonstrated a good performance, which can be important for a practical application, as it does not require document labels and pre-experimental click data.

8. CONCLUSIONS

In this paper, we stated the problem of the optimal scheduling of the online experiments. We described an experimentation pipeline model and formulated the optimal scheduling problem. Next, we introduced a greedy scheduling algorithm that ranks experiments according to their predicted probability of success. This algorithm allowed us to reduce the scheduling problem to a learning-to-rank problem. We studied pointwise and pairwise formulations of this learning-to-rank problem. To obtain a feature representation of the experiments, we considered relevance-based, click model-based, and exploration features. Finally, we performed a thorough evaluation study, examining how our proposed approaches compare to each other and to the baselines in terms of the prediction quality and the quality of the resulting schedules. We used two datasets that contain interleaving and A/B test experiments.

Our findings suggest that our proposed machine-learned schedule optimisation algorithms outperform the randomised, "natural" schedule by up to 342% when the number of the successful experiments performed under a limited number of available user interactions is measured on the dataset of the interleaving experiments. We also showed that a simple scheduler that ranks experiments according to their ranking effectiveness can achieve a smaller, but still a considerable improvement over the "natural" random baseline (up to 144% on the interleaving dataset, and up to 43% on A/B tests). Our study also suggests that an exploration-based scheduler can achieve a considerable improvement over an unoptimised schedule (43% improvement on the A/B tests dataset, 118% on the dataset of interleaving experiments). Notably, this can by applied for experiments where neither relevance judgements, nor historical click data is available.

An interesting direction for future work is to develop a two-stage scheduler, which uses offline-available features to pre-select experiments for exploration.

9. REFERENCES

[1] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM TOIS*, 30(1):6, 2012.

[2] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM 2009*.

[3] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW 2009*.

[4] K. Hofmann, S. Whiteson, and M. de Rijke. Estimating interleaved comparison outcomes from historical click data. In *CIKM 2012*.

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

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

[7] T. Joachims. Optimizing search engines using clickthrough data. In *KDD 2002*.

[8] T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*. Physica/Springer Verlag, 2003.

[9] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis. Using historical click data to increase interleaving sensitivity. In *CIKM 2013*.

[10] R. Kohavi, T. Crook, R. Longbotham, B. Frasca, R. Henne, J. L. Ferres, and T. Melamed. Online experimentation at Microsoft. *Data Mining Case Studies*, page 11, 2009.

[11] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu. Trustworthy online controlled experiments: Five puzzling outcomes explained. In *KDD 2012*.

[12] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *KDD 2013*.

[13] J. Y. Leung. *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, 2004.

[14] L. Li, J. Y. Kim, and I. Zitouni. Toward predicting the outcome of an A/B experiment for search relevance. In *WSDM 2015*.

[15] C. X. Ling, J. Huang, and H. Zhang. AUC: a statistically consistent and more discriminating measure than accuracy. In *IJCAI 2003*.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR 2011*.

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

[18] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM 2013*.

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

[20] G. Ridgeway. The GBM package. *R Foundation for Statistical Computing, Vienna, Austria*, 2004.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[22] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer. Overlapping experiment infrastructure: More, better, faster experimentation. In *KDD 2010*.

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