Using Interaction Data for Improving the Offline and Online Evaluation of Search Engines

Evgeny Kharitonov

School of Computing Science

University of Glasgow

A thesis submitted for the degree of

Doctor of Philosophy

May 2016

Abstract

This thesis investigates how web search evaluation can be improved using historical interaction data. Modern search engines combine offline and online evaluation approaches in a sequence of steps that a tested change needs to pass through to be accepted as an improvement and subsequently deployed. We refer to such a sequence of steps as an evaluation pipeline. In this thesis, we consider the evaluation pipeline to contain three sequential steps: an offline evaluation step, an online evaluation scheduling step, and an online evaluation step.

In this thesis we show that historical user interaction data can aid in improving the accuracy or efficiency of each of the steps of the web search evaluation pipeline. As a result of these improvements, the overall efficiency of the entire evaluation pipeline is increased.

Firstly, we investigate how user interaction data can be used to build accurate offline evaluation methods for query auto-completion mechanisms. We propose a family of offline evaluation metrics for query auto-completion that represents the effort the user has to spend in order to submit their query. The parameters of our proposed metrics are trained against a set of user interactions recorded in the search engine's query logs. From our experimental study, we observe that our proposed metrics are significantly more correlated with an online user satisfaction indicator than the metrics proposed in the existing literature. Hence, fewer changes will pass the offline evaluation step to be rejected after the online evaluation step. As a result, this would allow us to achieve a higher efficiency of the entire evaluation pipeline.

Secondly, we state the problem of the optimised scheduling of online experiments. We tackle this problem by considering a greedy scheduler that prioritises the evaluation queue according to the predicted likelihood of success of a particular experiment. This predictor is trained on a set of online experiments, and uses a diverse set of features to represent an online experiment. Our study demonstrates that a higher number of successful experiments per unit of time can be achieved by deploying such a scheduler on the second step of the evaluation pipeline. Consequently, we argue that the efficiency of the evaluation pipeline can be increased.

Next, to improve the efficiency of the online evaluation step, we propose the Generalised Team Draft interleaving framework. Generalised Team Draft considers both the interleaving policy (how often a particular combination of results is shown) and click scoring (how important each click is) as parameters in a data-driven optimisation of the interleaving sensitivity. Further, Generalised Team Draft is applicable beyond domains with a list-based representation of results, i.e. in domains with a grid-based representation, such as image search. Our study using datasets of interleaving experiments performed both in document and image search domains demonstrates that Generalised Team Draft achieves the highest sensitivity. A higher sensitivity indicates that the interleaving experiments can be deployed for a shorter period of time or use a smaller sample of users. Importantly, Generalised Team Draft optimises the interleaving parameters w.r.t. historical interaction data recorded in the interleaving experiments.

Finally, we propose to apply the sequential testing methods to reduce the mean deployment time for the interleaving experiments. We adapt two sequential tests for the interleaving experimentation. We demonstrate that one can achieve a significant decrease in experiment duration by using such sequential testing methods. The highest efficiency is achieved by the sequential tests that adjust their stopping thresholds using historical interaction data recorded in diagnostic experiments. Our further experimental study demonstrates that cumulative gains in the online experimentation efficiency can be achieved by combining the interleaving sensitivity optimisation approaches, including Generalised Team Draft, and the sequential testing approaches.

Overall, the central contributions of this thesis are the proposed approaches to improve the accuracy or efficiency of the steps of the evaluation pipeline: the offline evaluation frameworks for the query auto-completion, an approach for the optimised scheduling of online experiments, a general framework for the efficient online interleaving evaluation, and a sequential testing approach for the online search evaluation.

The experiments in this thesis are based on massive real-life datasets obtained from Yandex, a leading commercial search engine. These experiments demonstrate the potential of the proposed approaches to improve the efficiency of the evaluation pipeline.

Acknowledgements

This thesis culminates an exciting journey I started four years ago. It would not be possible to finish it without the immense support that I received over these years.

First and foremost, I must thank my supervisors, Iadh Ounis and Pavel Serdyukov, and my co-supervisor, Craig Macdonald. Their supervision has taught me how to build upon the existing work and how to tackle the scientific problems that I believe are important. Their insightful advice and feedback guided this journey.

I am thankful to my colleagues at the Terrier team and Yandex, for the collaboration, the support, and the sense of camaraderie they provided: Rodrygo Santos, Richard McCreadie, Nut Limsopatham, Dyaa Albakour, Romain Deveaud, Graham McDonald, Anjie Fang, Stuart Mackie, Aleksandr Vorobev, Aleksandr Chuklin, Artem Babenko, and Alexey Borisov. I have learned a lot from these amazing people, and I feel honoured to have worked with them.

I am grateful to Ilya Segalovich. His passion for science, curiosity, and energy were contagious and motivated me to start this journey. He truly inspired all of us who where lucky to know him. His untimely death left a void in all of his colleagues at Yandex.

I would like to thank my parents and my sister, who always encouraged my ambitions. Lastly, I am specially grateful to my beloved spouse, Tania, for her endless support and encouragement, despite the miles separating us.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The evaluation of search engines has always been a crucial part of Information Retrieval. Being equally important in research and industry applications, evaluation is used both to assess the validity of scientific hypotheses and to test improvements in commercial search engines.

Search evaluation has constantly evolved over time to reflect new applications and challenges. Historically, from the moment it was introduced in the 1960s, offline system-based evaluation methods[1] (Cleverdon, 1967; Voorhees, 2002) became widely accepted due to their repeatability, reproducibility, and interpretability. These methods form the foundation of evaluation initiatives such as the Text REtrieval Conference (TREC)[2], the Cross-Language Evaluation Forum (CLEF)[3], and the NII-NACSIS Test Collection for IR Systems (NTCIR)[4].

Despite being a highly successful tool, system-based methods are difficult to apply to evaluate changes in some modern search applications, such as personalised search, or in the overall user search experience.

Thus, between 2000 and 2010, new online evaluation methods started to attract attention. Indeed, the modern major web search engines serve billions of queries per day. This scale of operation provides the search engines with an affordable and convenient evaluation tool: the implicit feedback of their users. In online experiments, such as A/B tests and interleaving, the users experience a modified version of a search engine (Joachims, 2003; Kohavi *et al.*, 2009). After that, the effects of the changes on user behaviour are analysed. This evaluation process allows search engines to make informed decisions about their progress even in the cases when the offline evaluation methods are hard to apply.

[1] In this thesis, we use *system-based evaluation* and *offline evaluation* interchangeably.
[2] http://trec.nist.gov/
[3] http://www.clef-initiative.eu/track/series
[4] http://research.nii.ac.jp/ntcir/index-en.html

In turn, online evaluation methods have their limitations, too. They rely on implicit user feedback, which can be noisy and difficult to interpret. Hence, each experiment requires a considerable number of observations to make a statistically reliable conclusion. Usually, an online experiment spans a time period of one week or more (Chapelle *et al.*, 2012; Drutsa *et al.*, 2015; Kohavi *et al.*, 2012, 2013), and requires up to several per cent of the search engine's query traffic for each experiment (Kohavi *et al.*, 2012, 2013). These constraints considerably restrict the usefulness of the online evaluation methods.

We use the term *evaluation pipeline* to describe a sequence of steps that a tested change in a search engine needs to pass through to be accepted as an improvement and subsequently deployed. An important requirement of an evaluation pipeline is its *efficiency*. Informally, we consider the evaluation pipeline as more efficient, if a higher number of successful changes are obtained in a unit of time. We consider that the evaluation pipeline involved in testing a new approach contains three steps: an offline evaluation step (involving classical test collections and offline evaluation measures, to determine a priori how effective the approach is), a scheduling step (to decide which approach should next be evaluated using online experiments), and finally the online evaluation step (where online A/B and interleaving experiments validate the user experience under the new approach). These steps are discussed in more detail in Chapter 3.

How to use the existing evaluation methods to organise the evaluation pipeline, and how to make this pipeline efficient without reducing its reliability, is an important scientific problem, which we approach in this thesis. We study how the entire evaluation process, including offline and online steps, can be organised, and investigate how these steps can be improved. In particular, we study how to make the results of the offline evaluation better aligned with online evaluation. Further, we investigate how to organise the online evaluation of the search engine's changes and how to optimise the online experiments so that the results are obtained faster.

The historical interaction logs of the search engine's users are an invaluable source of data. Numerous studies discuss how user interaction data can be used in various tasks, ranging from building query auto-completion mechanisms (Bar-Yossef & Kraus, 2011) to contextualised ranking (Shokouhi *et al.*, 2013) and offline evaluation of personalised search results (Bennett *et al.*, 2011; Kharitonov & Serdyukov, 2012). In this thesis the historical user feedback data is the main tool for improving the evaluation pipeline.

For instance, the historical interaction data can be used to build a model of the user's interactions with the query auto-completion mechanism. This model can be used to predict how the users will interact with a new ranking of query completions, and, in turn, if this new ranking is better than the one used earlier. As a result, this model can be used as a foundation for an offline metric that is highly

correlated with the outcomes of online experiments, which evaluate new mechanisms by exposing them to real users. Thus, a higher proportion of the ranking algorithms that have been fine-tuned to improve the offline metric will demonstrate improved user satisfaction in the online experiments. As a result, the efficiency of the evaluation pipeline increases.

In another example, the historical interaction data can be used to improve the sensitivity of the online experiments (Chapelle *et al.*, 2012; Yue *et al.*, 2010). This can be achieved by adjusting the parameters of the online experimentation method so that the confidence in the earlier performed experiments is maximised. As a result, the convergence rate of future experiments can be increased. In turn, a higher convergence rate implies that experiments can be deployed for a shorter period of time, thus improving the efficiency of the evaluation pipeline.

The above two examples illustrate how the historical user interaction data can be leveraged to improve the efficiency of the evaluation pipeline, and, as we discuss in Section 1.3, this idea is central to this thesis.

In the remainder of this chapter we discuss the motivation for the work in this thesis, and present the statement of this thesis and its contributions. We close this chapter with an overview of the structure of this thesis.

1.2 Motivation

Modern commercial web search engines are extremely sophisticated systems, developed by thousands of engineers. Making informed decisions while developing these systems is a non-trivial task. Indeed, when working on such a scale, even tiny changes in the user search experience might have a considerable impact on the search engine's revenue and market share. For instance, it was reported that changes in the colours of the result links can lead to millions of dollars of revenue gains due to users clicking on ads more often.[5] Changes in the search engine's response time even smaller than 500 ms can badly affect the user (Barreda-Ángeles *et al.*, 2015; Kohavi *et al.*, 2013). These examples highlight the necessity to thoroughly evaluate the majority of the deployed changes to the search engine.

This gave rise to the data-centric culture popularised by Google[6] (Tang *et al.*, 2010) and Bing (Kohavi *et al.*, 2013). Under this data-centric approach, all changes to the search engine are tested and

[5] `http://www.lukew.com/ff/entry.asp?1025`: "...Microsoft also tested multiple versions of blue for links in their search results. A specific color of blue (#0044CC) drove $80-$100 million dollars a year increase over the light blue the design team tried first". This story is also reported in (Kohavi *et al.*, 2014).

[6] "At Google, experimentation is practically a mantra" (Tang *et al.*, 2010).

all decisions are supported by observational data. A similar approach is widely used by other Internet industry leaders such as Amazon[7] and Netflix[8].

The rise of this data-centric culture imposed considerable challenges on Information Retrieval evaluation methods. Indeed, when the scale and the speed of the search engine development increases, the stream of changes to be evaluated grows as well. For instance, it was reported that the number of simultaneously deployed online experiments at Bing grew almost exponentially in time and reached the order of several hundreds of experiments deployed at any given moment (Kohavi *et al.*, 2013).

Under such a scenario, new requirements for the efficiency of the web search evaluation arise. Indeed, the speed of the evaluation bounds the rate of the search engine's evaluation: the faster a change can be tested, the faster it can be deployed or rejected and returned for further development. From this point of view, the evaluation efficiency becomes a competitive advantage on the search engine market. At the same time it is crucial not to reduce the accuracy of the evaluation. Both a rejected improvement and a deployed feature that does not improve users' satisfaction could lead to high costs for a search engine.

Overall, these observations lead us to the following question: how can the efficiency of the web search evaluation pipeline be improved? This question is central to this thesis and in the next sections we discuss our approach to address it.

1.3 Thesis Statement

This thesis states that historical user interaction data can aid in improving the accuracy or efficiency of each of the steps of the web search evaluation pipeline. In particular, we hypothesise that user interaction data can be used to devise novel offline metrics for the evaluation of query auto-completion mechanisms that are aligned with the user online behaviour. Furthermore, we hypothesise that the scheduling of the online experiments can be improved by using the historical click data. We also argue that user interaction data can be leveraged to improve the efficiency of interleaving experiments, and extend their applicability to new domains, such as image search. Finally, we hypothesise that the user interaction data can be used to develop efficient statistical analysis procedures for the online experimentation, so that the experiments can be stopped earlier, thus reducing the average duration of the experiments and increasing their efficiency.

[7] http://ai.stanford.edu/~ronnyk/emetricsAmazon.pdf
[8] http://techblog.netflix.com/2014/08/scaling-ab-testing-on-netflixcom-with_18.html

1.4 Contributions

In this thesis, we firstly describe a search engine's evaluation pipeline that combines both offline and online evaluation in a single pipeline that is used for effective data-driven search engine evaluation. We split this pipeline in several steps, including the offline step, the online experiment scheduling step, and the online evaluation step. Each of these components has a considerable impact on the pipeline efficiency. For each of these steps, we propose and evaluate novel approaches to improve or extend such steps. Our proposed approaches are shown to improve the evaluation pipeline across three domains: query auto-completions, document search, and image search. The main contributions of this thesis are:

Offline Evaluation We propose a family of offline query auto-completion metrics. These metrics are based on a model of the user behaviour that is trained using historical user interaction data. We experimentally demonstrate that these metrics are better aligned with the results of online experiments than the previously used metrics.

Scheduling Scheduling is the second step of our considered evaluation pipeline. We propose to improve its efficiency by addressing the problem of optimised scheduling of the online experiments. More specifically, we compare the quality of the schedules produced by a set of offline and online predictors of the experiment's success, and their machine-learned combinations. Our study demonstrates that a higher number of successful experiments per unit of time can be achieved by using a learning-to-rank based combination of the studied predictors, thus increasing the evaluation efficiency.

Online Evaluation Our contributions to improving the efficiency of the online evaluation step are three-fold. Firstly, we investigate how the sensitivity of the interleaving experiments can be improved. We consider two approaches to address this problem. In the first approach, we propose to optimise the interleaving sensitivity by selecting an appropriate per-query interleaving policy.

In the second approach, we develop the Generalised Team Draft interleaving framework, that generalises the existing Team Draft algorithm to domains with a grid-based result representation (e.g. image search) and combines several approaches for the sensitivity optimisation. Specifically, Generalised Team Draft interleaving optimises how each click is weighted and how often each possible interleaved result page is shown to achieve a higher sensitivity. Importantly, the parameter optimisation is performed with respect to the historical online experimental data, containing a vast number of user interactions. In addition, Generalised Team Draft uses a stratified outcome estimator that can also increase the interleaving sensitivity in some cases.

In the third part of our contribution, we study the usefulness of the sequential statistical tests that are capable of stopping online experiments when the collected data is sufficient to make reliable con-

clusions for increasing the efficiency of interleaving online experiments. We propose two modifications of the existing sequential tests that can be applied in the search evaluation scenario. We describe how historical experimental data can be used to adjust the stopping thresholds for these tests. We perform an evaluation study, assessing the usefulness of the sequential testing approach. Finally, we demonstrate that the interleaving sensitivity optimisation approaches, such as Generalised Team Draft, and sequential statistical testing can be combined together to achieve even higher efficiency gains.

The experiments conducted in this thesis are performed on real-life datasets obtained from Yandex, one of the world's major search engines. These datasets contain millions of user interactions. Some of the datasets (e.g. the dataset used in Chapter 6) span almost a year worth of experiments deployed by Yandex.

1.5 Origins of Material

The material in this thesis is based on a number of conference publications:

- Chapter 4: The offline machine-learned query auto-completion metrics we discuss in Chapter 4 were initially proposed in our SIGIR 2013 publication (Kharitonov *et al.*, 2013*b*).
- Chapter 5: Our proposed framework to optimise the schedule of the queue of the online experiments was initially published in SIGIR 2015 (Kharitonov, Macdonald, Serdyukov & Ounis, 2015*b*).
- Chapter 6: The interleaving sensitivity optimisation is discussed in two of our papers. The first work (Kharitonov *et al.*, 2013*c*) was published in CIKM 2013, and it discusses how the historical clickthrough data can be used to improve the interleaving sensitivity (Section 6.3). The second paper (Kharitonov, Macdonald, Serdyukov & Ounis, 2015*a*), published in CIKM 2015, introduces our Generalised Team Draft interleaving framework. This framework is discussed in Section 6.4.
- Chapter 7: Our study of the sequential significance testing algorithms as a tool to increase the efficiency of the online experimentation and our proposed sequential tests are based on the SIGIR 2015 publication[9] (Kharitonov, Vorobyev, Macdonald, Serdyukov & Ounis, 2015).

Finally, the need for the image search-based interleaving algorithm (Chapter 6) was argued in the Doctoral Consortium work (Kharitonov, 2014).

[9]This paper received the SIGIR 2015 Best Student Paper award.

1.6 Thesis Outline

The remainder of this thesis is organised as follows:

- In Chapter 2 we introduce the background essential for this thesis. Specifically, we start with an overview of the existing evaluation approaches: offline and online. We discuss the Cranfield evaluation paradigm and the variety of the existing offline evaluation metrics that have been used to assess the performance of information retrieval systems in Section 2.2. In Section 2.3, we discuss state-of-the-art existing online evaluation approaches, such as A/B testing and interleaving.

- In Chapter 3, we discuss a typical evaluation pipeline that combines the existing evaluation approaches in a single decision process. This discussion provides us with a roadmap to improving the efficiency of the evaluation pipeline as a whole by improving its individual steps.

- Chapter 4 discusses how the offline evaluation step of the pipeline can be improved in the case of the query auto-completion evaluation. We study how novel offline metrics for the query auto-completion mechanisms can be constructed so that they are better aligned with the online behaviour of users.

- Chapter 5 investigates how the efficiency of the experimentation pipeline can be increased by the improved scheduling of the queue of the online experiments. We discuss different approaches to address this problem, and empirically compare them on a dataset of interleaving experiments.

- In Chapter 6, we study how the efficiency of the online evaluation experiments can be improved by increasing the sensitivity of interleaving. We investigate two approaches to improve the interleaving sensitivity. First, we discuss a possible click model-based approach to increasing the interleaving sensitivity in Section 6.3, as well as its limitations. Further, in Section 6.4 we tackle the same problem from a different perspective and address the limitations of our earlier approach. Specifically, we investigate how an increased sensitivity can be achieved by combining machine-learned credit assignment functions, result page-based stratification, and the interleaving policy optimisation.

- Chapter 7 discusses how the sequential statistical tests can be used to improve the efficiency of the online experimentation, and introduces our modification of the MaxSPRT sequential test for interleaving.

- In Chapter 8, we study how different approaches to improve the efficiency of the online evaluation can be combined together. In this chapter we focus on evaluating the combined improvement of the interleaving efficiency by jointly applying the sensitivity optimisation and the sequential testing approaches.
- Chapter 9 closes this thesis by highlighting the contributions and the conclusions drawn from each of the individual chapters. Finally, we discuss possible research directions for future work.

Chapter 2

Background

2.1 Introduction

In this thesis we consider the problem of evaluation as a pairwise comparison problem: given two systems, A and B, can we predict which of them is more likely to satisfy the users? These systems might be two different search engines (e.g. Bing vs. Google), or two variants of the same system (e.g. the current production system vs. the same system with the ranking algorithm changed). A variety of approaches are used in Information Retrieval to address this problem. Since the 1960s, the offline system-based evaluation approach gained popularity among Information Retrieval researchers due to its convenience (e.g. it does not require a search system to have any users) and reproducibility. On the other hand, the rise of the massively popular online search engines in the 2000s permitted progress beyond the abstractions assumed by the offline approach, and directly measure satisfaction indicators by exposing the search engine's users to the tested changes.

In this chapter, we review both approaches. In Section 2.2, we discuss the system-based evaluation, and in Section 2.3 we review the online evaluation approaches. We briefly compare offline and online evaluation approaches in Section 2.4. In Section 2.5, we provide a focused review of the click models used in this thesis. We conclude this chapter with Section 2.6.

Overall, it is almost impossible to provide an in-depth overview of all topics in modern web search evaluation. Instead, in this chapter we aim to get a high-level overview of the web search evaluation landscape. In each technical chapter of this thesis we will include a more specialised background section, discussing the work that is related to this chapter specifically.

2.2 Offline Evaluation

The foundation of the modern system-based evaluation was laid in the Cranfield experiment (Cleverdon, 1967; Voorhees, 2002). The original Cranfield experiment was designed to isolate the effectiveness of the evaluated systems from all other variables and to evaluate it separately. The central concept of the Cranfield evaluation paradigm is the test collection. Such a collection contains a set of documents, a set of queries (often referred to as topics), and a set of relevance judgements for query-document pairs (these judgements are also called labels).

Given a test collection, the evaluation is performed as follows. Firstly, for each query in the test collection, the ranked lists of results are obtained from the evaluated systems. After that, each ranking is associated with a score that represents a value of an effectiveness metric, such as MAP (Manning *et al.*, 2008), ERR (Chapelle *et al.*, 2009) or nDCG (Järvelin & Kekäläinen, 2002). Usually, metrics favour rankings with relevant documents closer to the top of the list and we discuss them in Section 2.2.1.

Selecting a set of queries for evaluation is not a trivial task. This set must reflect the actual stream of queries for the evaluation to be meaningful, so usually real queries from the query stream are used. The selected set of queries should be large enough to be able to differentiate the compared systems. At the same time, the number of the queries used determines the amount of effort required to label the documents, hence the cost of building the test collection. Industrial test collections can contain thousands of queries (Chapelle *et al.*, 2009), while the TREC evaluation collections use 50 queries (Clarke *et al.*, 2009; Collins-Thompson *et al.*, 2014). In order to achieve a higher discrimination rate between the compared systems with the restricted number of queries, in TREC (Clarke *et al.*, 2009; Collins-Thompson *et al.*, 2013) the used queries are selected to be less frequent and possibly more "difficult" for the evaluated systems, hence more discriminative.

After a set of queries is selected, the next step is to judge the documents' relevance w.r.t. the selected queries. In large-scale applications, such as web document search, it is clearly impossible to label all documents a search engine is aware of. For this reason, for each query, a set of documents to be labelled is pre-filtered. This process is referred to as *pooling* (Sanderson, 2010; Sparck Jones & van Rijsbergen, 1975). In the simplest case, each query is submitted to all the evaluated systems, and their results are retrieved. After that, the documents that are ranked in top-k results by at least one of the evaluated systems are merged in a single set. For instance, in early TREC campaigns the union of the top-100 documents retrieved by the evaluated systems were judged for each topic (Sanderson, 2010). This form of pooling assumes that if a document is not ranked high by any of the evaluated systems, it is unlikely to be relevant.

Instead of simply selecting top-k ranked documents, one can use more strategic pooling methods, e.g. (Cormack *et al.*, 1998; Moffat *et al.*, 2007). For instance, Moffat *et al.* (2007) studied the pooling strategies that adaptively change priority of a document in the labelling queue based on how useful it would be to have it labelled. Notably, the estimate of a document's usefulness changes when other documents are labelled.

After the sets of queries and documents are established, the last step for building a test collection is to obtain the relevance labels or judgements. The relevance judgements are usually obtained by the relevance judges in a manual labelling process. In this process, each judge is presented with a query and a document, and after examining them, produces a relevance label. This label reflects a degree of relevance of the document w.r.t. this particular query.

Due to being manual labour, the labelling process tends to be expensive and time-consuming. The judges can be trained professionals, as in TREC (Clarke *et al.*, 2009), or participants of some crowd-sourcing platform (Alonso *et al.*, 2008). In both cases, the judges can be subjective and are prone to errors. For instance, a judge can misinterpret the query "rio", considering it to be related to the animation film, while the user who submitted it was looking for information about the city. While one might argue that subjectivity can be actually useful, as the tastes and query interpretations of the search engine's users can differ (Manning *et al.*, 2008), search engines use some form of guidelines or training for judges (Megorskaya *et al.*, 2015). In particular, this can be required if the judgements have more than two grades (discussed in Section 2.2.1), as used by search engines and TREC (Clarke *et al.*, 2009; Sanderson, 2010).

Measuring the quality of relevance labels is a hard task, since there usually are no true labels. Often, inter-judge agreement is used as a proxy for such a quality. The *kappa* statistic is used to measure the level of agreement (Manning *et al.*, 2008). This statistic characterises how often the judges actually agree in their labels in comparison with case when they agree solely by chance. In an alternative approach, Megorskaya *et al.* (2015) used labels produced by highly experienced judges as ground-truth labels.

Once a test collection is built, it can be repeatedly used to evaluate new systems with little or no additional cost. Moreover, the results that are obtained on the same test collection by independent researchers can be meaningfully compared. These specifics explains the high popularity of the system-based approach.

However, given a test collection, a question arises about how to measure the effectiveness of a search engine. To address this question a variety of offline effectiveness metrics have been considered, and we discuss them in more details in the next section.

Table 2.1: Definitions of the quantities calculated for a retrieved list, (Cleverdon & Keen, 1966; Kent *et al.*, 1955; Sanderson, 2010). For instance, a is the number of relevant documents retrieved by the evaluated system.

	Relevant	Not Relevant
Retrieved	a	b
Not Retrieved	c	d

2.2.1 Effectiveness Metrics

An offline effectiveness metric is used to numerically represent the quality of the ranking. Essentially, the test collection in combination with an evaluation metric *simulates* users using a search system (Sanderson, 2010). Since early search systems returned all documents that match the query, the evaluation measures matched this usage scenario (Sanderson, 2010). As a result, the effectiveness metrics such as *Precision* and *Recall* were introduced (Kent *et al.*, 1955) that rely on the binary relevance labels.

For a particular query, Precision is defined as a ratio of the relevant documents retrieved by a system and the total number of documents retrieved. Similarly, Recall is defined as the fraction of the relevant documents retrieved by the system. More formally, using the definitions from Table 2.1, Precision $P(q)$ and Recall $R(q)$ for the query q can be computed as follows (Kent *et al.*, 1955):

$$P(q) = \frac{a}{a+b}, \qquad R(q) = \frac{a}{a+c}$$

Later, Van Rijsbergen (1974) proposed a metric that equates to one minus the weighted harmonic mean of recall and precision. More often the weighted harmonic mean metric is used (Sanderson, 2010), and it is referred to as F-measure. Several equivalent formulations of the F-measure metric exist (Croft *et al.*, 2010). For instance, it can be calculated as follows:

$$F_\beta(q) = (1+\beta^2) \cdot \frac{P(q) \cdot R(q)}{\beta^2 \cdot P(q) + R(q)} \tag{2.1}$$

In Equation (2.1), the parameter β specifies the relative importance of the Precision over recall. If $\beta > 1$, then F_β considers Precision to be more important. Sometimes, $F_{0.5}$ and F_2 metrics are used, however, F_1 is the most frequently used in the literature. Under F_1 Precision and Recall are equally important.

The choice of the harmonic mean instead of the arithmetic mean allows F-measure to stronger penalise systems with low Recall or low Precision. Indeed, if a system has Recall equal to 1 and Precision approaches 0, the arithmetic mean of these scores would be 0.5, while the harmonic mean is equal to 0. This intuition can be extended, as the harmonic mean of a finite set of positive numbers is always smaller or equal than the arithmetic mean of the same set. As a result, if starting from a point where

Precision and Recall are equal (and equal to their arithmetic and harmonic means), and increasing Recall (Precision) with Precision (Recall) fixed, we will observe that the harmonic mean grows slower and remains closer to the starting point.

The above introduced metrics are defined for the evaluation of the Boolean retrieval systems that only return a set of documents that they consider to be relevant. However, as ranked retrieval systems developed (Sanderson, 2010), the need for their evaluation appeared.

At first, this need was addressed by measuring Precision at pre-specified Recall levels (e.g. 10%, 20%, ... , 90%) with some form of interpolation (Cleverdon & Keen, 1966). Clearly, in the context of modern huge test collections[10] and Internet search engines, it is not practical to assume that all relevant documents can be known for every query in the test collection, thus it is hard to operate with the Recall levels. Similarly, it is less likely that a user will examine a low-ranked result (e.g. placed on the 10th page). For this reason, Precision at a fixed cut-off level n was introduced:[11]

$$P@n(q) = \frac{a}{n}$$

The TREC evaluation campaign made the Precision at rank n, Mean Average Precision (MAP) (Harman, 1994), R-precision (Harman, 1994), and Mean Reciprocal Rank (MRR) (Kantor & Voorhees, 1996) metrics popular. Along with providing standard test collections, TREC provided an implementation of various offline evaluation metrics in the form of the *trec_eval* software, which is now considered as a de-facto standard implementation of the metrics.[12]

Later, both industrial and academic researchers (Sanderson, 2010) started to use graded (non-binary) relevance labels. At the moment, a five-point scale is typically used (Carterette & Jones, 2008; Chapelle *et al.*, 2011): {Perfect, Excellent, Good, Fair, Bad}. However, the above discussed metrics are not able to fully leverage these relevance levels, and, as a result, this gave rise to the metrics such as Cumulated Gain (CG), and Discounted Cumulated Gain (DCG) (Järvelin & Kekäläinen, 2000).

CG is defined as a sum of the relevance grades of the top n documents:

$$CG = \sum_{i=1}^{n} rel(i)$$

where $rel(i)$ maps the relevance grade i to a numeric representation, e.g. *Perfect* is mapped to 5, *Excellent* corresponds to 4, etc. Discounted Cumulative Gain additionally penalises gains from relevant

[10] For instance, ClueWeb12 contains 733,019,372 English web pages, `http://www.lemurproject.org/clueweb12.php/`.

[11] The cut-off level used varies depending on the result representation used. Usually, the cut-off level of 10 is used for web document search.

[12] `http://trec.nist.gov/trec_eval/`

documents appearing later in the ranking, to some extent simulating the user's attention model:

$$DCG = rel(1) + \sum_{i=2}^{n} \frac{rel(i)}{\log_2(i)} \tag{2.2}$$

Another variant of DCG is also commonly used:

$$DCG = \sum_{i=1}^{n} \frac{2^{rel(i)} - 1}{\log_2(i+1)} \tag{2.3}$$

In contrast to Equation (2.2), the definition in Equation (2.3) emphasises retrieving highly relevant documents. This form is used by some search engines (Croft *et al.*, 2010).

A normalised version of DCG (nDCG) was also considered (Järvelin & Kekäläinen, 2002). The motivation behind this metric is to balance the impact of different queries on the overall effectiveness score. Indeed, some queries have numerous highly relevant results, and typically their DCG scores are high. In contrast, some queries can only have few relevant documents, and it is impossible to gain a high score for these queries. The nDCG metric balances these per-query scores by normalising them to the score of the ideal ranking:

$$nDCG = \frac{DCG}{ideal \quad DCG}, \quad 0 \leq nDCG \leq 1$$

Later, Moffat & Zobel (2008) argued that the logarithmic decay used in the DCG-like metrics does not perfectly model the user behaviour when examining a list of ranked documents, and proposed the Rank Biased Precision (RBP) metric, which models the list examination process assuming a geometric discount:

$$RBP = (1-p) \cdot \sum_{i=1}^{n} rel(i) \cdot p^{i-1}$$

with p denoting the probability that the user progresses to the next documents, and n is the considered cut-off level. The value of the metric equates to the expected cumulative relevance of the results that the user examines in the process where after each result the user stops with probability $(1-p)$. Higher values of p model a persistent user, and low values of p correspond to a less persistent user.

The geometric discount with p set to 0.7 closely approximates the position examination probabilities (Chapelle *et al.*, 2009). However, it totally ignores dependencies between the examination events. Indeed, the examination of the second document is less likely to occur if the first document is highly relevant and completely satisfies the user. Similarly, the user might be more likely to examine the second result if the first is non-relevant. This effect is modelled in cascade-based models of clicking behaviour (Chapelle & Zhang, 2009; Craswell *et al.*, 2008), which we discuss in more detail in Section 2.5.

The cascade model proposed in (Craswell *et al.*, 2008) was used as a foundation to the Expected Reciprocal Rank (ERR) metric (Chapelle *et al.*, 2009). ERR and its modifications were the metric of choice in the TREC Web Track evaluation campaign over several years (Collins-Thompson *et al.*, 2013, 2014).

ERR is defined as an expectation of the reciprocal function $\phi(i) = \frac{1}{i}$ at the position i where the user stops their examination process. This position is calculated based on the cascade model which is discussed in details in Section 2.5. Under this model, the user examines the result list from top to bottom. A document in ith position satisfies the user with a probability P_i that depends on the document's relevance grade $rel(i)$. Denoting the maximal relevance grade as rel_{max}, this probability is calculated as follows:

$$P_i = \frac{2^{rel(i)} - 1}{2^{rel_{max}}} \tag{2.4}$$

A satisfied user stops their search, otherwise they continue examining the result page. Having these considerations in mind, we calculate the expectation of $\phi(i)$ as follows:

$$ERR = \sum_{i=1}^{n} \frac{1}{i} \cdot P_i \prod_{j=1}^{i-1} (1 - P_j) \tag{2.5}$$

ERR and RBP were among the first metrics that are explicitly build on top of the user behaviour modelling ideas. This idea was further extended by Yilmaz *et al.* (2010) and Chuklin, Serdyukov & de Rijke (2013).

Overall, the above focused review of offline evaluation in this chapter demonstrates that the recent developments are concentrated on improving our understanding and increasingly accurate modelling of the user interactions with the ranked result lists. This improved modelling results in better metrics: the logarithmic decay in the DCG-based metrics is replaced with the geometric decay in RBP, the simple user model used in RBP is replaced by an improved model in ERR, etc.

Our work in Chapter 4 continues this line of work, and leverages an improved modelling of user interactions with query auto-completion mechanisms to build an evaluation metric that is aligned with user behaviour.

2.3 Online Evaluation

Returning to the evaluation problem discussed in Section 2.1, we need to compare two systems, A and B, to infer which of them is more likely to better satisfy the users. In the offline evaluation approach discussed in Section 2.2, we use a test collection to simulate an operational setting of both alternatives and measure their effectiveness by means of offline evaluation metrics. In contrast, in online evaluation a direct observation is used: the users experience a modified version of the search engine and the

changes in their behaviour (if any) are analysed. In this section, we consider the two most popular online evaluation approaches: A/B testing (Section 2.3.1) and interleaving (Section 2.3.2).

A closely related concept to the evaluation efficiency is the *sensitivity* of online experiments. We typically define the sensitivity as the ability to obtain experiment outcomes at a pre-specified confidence level with as few observations as possible. A higher sensitivity implies higher efficiency of the online evaluation, and can be considered as the convergence speed of the evaluation metric in an online evaluation experiment.[13] It was demonstrated that interleaving evaluation is more sensitive than A/B tests (Chapelle *et al.*, 2012; Schuth *et al.*, 2015).

2.3.1 A/B testing

The most straightforward way to infer if the users are going to like a new system is to deploy this system and use it to serve the queries of a sample of users and observe changes in their behaviour. In order to exclude a possibility of the changes occurring due to external factors (e.g. temporal changes in the traffic), usually another random set of users is also observed. The queries of the users from the latter group are served by the unchanged, production system. Assuming that these two groups of users are samples from the same user population and the only difference in the search engine variants they are using is due to the experimental change, it is safe to conclude that any statistically significant difference in the user behaviour is due to the tested change.

In Information Retrieval, this method is referred to as an A/B test. However, similar controlled randomised tests were used for a long time in the clinical trials practice. For this reason, the group of the users that are served by the changed system are referred to as the *treatment* group, while the second group is called the *control* group. These groups are denoted as B (treatment) and A (control).

To quantify the level of the users' satisfaction with the tested systems, some *online absolute metrics* are measured for the control and treatment groups. In the simplest case, the abandonment rate[14] can be used as such a metric (Radlinski *et al.*, 2008). A variety of more elaborated metrics were studied in the literature. In general, these absolute metrics can be divided in three groups.

The first group of absolute metrics operate on the level of a single *search engine result page* (SERP). These metrics try to quantify the users satisfaction with the results by analysing their behaviour on the result page. The abandonment rate, the mean reciprocal rank of the clicked results (Radlinski *et al.*, 2008), the time between submitting the query and the first click (Radlinski *et al.*, 2008) are examples

[13] This concept is related to estimator efficiency used in statistics (Cramér, 1946).
[14] How often a result page is not clicked by a user.

of the SERP-level metrics. The SERP-level metrics are among the most sensitive, however, the interpretation of these metrics might be not straightforward. For instance, the treatment group having lower abandonment rate can be a good sign (users find relevant results more often) and a bad sign (e.g. one of the verticals[15] stopped providing a satisfactory answer on the result page and users need to click on the results).

The second group of absolute metrics proposed in the literature operates on the session level. The pSwitch metric (Arkhipova *et al.*, 2015) and the user search goal success predictor proposed by Hassan *et al.* (2010) are examples of metrics from this group. These metrics analyse the user behaviour that spans for an entire session and quantify how successful this session was. For instance, if a session ended by the user switching to a competing search engine, then it is likely that the search engine did not satisfy the user. On one hand, these metrics can be more interpretable than the SERP-level metrics. On the other hand, these metrics are the outputs of session success classifiers, which are essentially machine-learned black boxes. Moreover, these classifiers might require to be re-trained after some period of time, as the characteristics of user behaviour change over time, e.g. when changes in UI occur.

The last group of metrics include the absolute metrics that quantify the user behaviour over a span of many sessions. These metrics include sessions per user (Kohavi *et al.*, 2012), absence time (Chakraborty *et al.*, 2014; Dupret & Lalmas, 2013), and some others (Drutsa *et al.*, 2015). These metrics are a better proxy to the quantities that are of direct interest for commercial search engines, such as the market share or the time users spend on the service.

Once the metric of interest is fixed, it is calculated for both the control and the treatment groups. In this thesis, the means of the absolute online metrics for alternatives A and B are denoted as μ_A and μ_B. Due to the intrinsic random noise in user behaviour, the means of the selected metric are always different in the control and treatment groups. Thus, a statistical significance test is used to determine if the available data is sufficient to infer any preference over the tested alternatives A and B, as described in Section 2.3.3.

2.3.2 Interleaving

In contrast to A/B tests, interleaving-based evaluation operates on the ranking list level. Thus it can only be used to evaluate the relative quality of the rankers, and is not applicable to compare changes in the user interface, for instance. Similarly to A/B testing, we denote these rankers as A (the production system) and B (the tested alternative ranker).

[15] Some search engines incorporate results from specialised search services, called verticals, into their Web search result pages (Arguello *et al.*, 2009; Diaz, 2009). *News* and *Images* are examples of such verticals.

Input: Rankings $A = (a_1, a_2, ...)$ and $B = (b_1, b_2, ...)$
Output: Interleaved ranked list L
Initialised pointers to the results in rankings A and B
$k_a \leftarrow 0$
$k_b \leftarrow 0$
if $k_a == k_b$ **then**
  if $A[k_a + 1] \notin L$ **then**
    $L \leftarrow L + A[k_a + 1]$
  end
  $k_a \leftarrow k_a + 1$
else
  if $B[k_b + 1] \notin L$ **then**
    $L \leftarrow L + B[k_b + 1]$
  end
  $k_b \leftarrow k_b + 1$
end

Algorithm 2.1: Balanced Interleaving algorithm used to interleave two result lists, introduced by Joachims (2002). To avoid a systematic bias, Joachims (2002) proposed randomise the ranking to start with. This algorithm provides steps for the case when interleaving starts with ranking A.

In an interleaving experiment, for every query a user submits both rankings from A and B are obtained. After that these results are *interleaved* (i.e. mixed) in a randomised manner and shown to the user. The clicks of the users are collected over a period of time (e.g. several days) and analysed afterwards.

A variety of interleaving algorithms exist. The first interleaving algorithm, Balanced Interleaving, was proposed by Joachims (2002). Later, Team Draft (Radlinski *et al.*, 2008), Probabilistic Interleave (Hofmann *et al.*, 2011), Optimized Interleaving (Radlinski & Craswell, 2013), and their modifications were introduced. These methods differ in the way the interleaved result pages are generated and how user click feedback is interpreted.

Under Balanced Interleaving (Joachims, 2002), the interleaved result page is built by running Algorithm 2.1. Alternatives A and B are associated with pointers to the results, k_a and k_b, respectively. These pointers iterate over the ranked results and are used to select the next result to be included in the combined result list. This process guarantees that at each cut-off level the number of documents contributed by A and B differ by at most 1. After the combined result page is generated, it is shown to the user and the alternatives are attributed with clicks according to how many clicked documents each of the alternative rankings retrieved in their top results. However, one can construct an example pair of rankings (often called *a breaking case*) that illustrates that a randomly clicking user can induce a preference towards one of the alternatives (Radlinski *et al.*, 2008). This indicates that Balanced Interleaving might lead to biased evaluation in some cases.

In contrast to Balanced Interleaving, Team Draft (Radlinski *et al.*, 2008) behaves correctly under the breaking case considered in (Radlinski *et al.*, 2008). However, one can still construct an example where Team Draft is biased (Chapelle *et al.*, 2012; Hofmann *et al.*, 2011). While it is not clear if these query-level breaking cases can affect system-level comparisons, it is clearly better to avoid these biases. In Optimised Interleaving proposed by Radlinski & Craswell (2013), a formal unbiasedness criterion is introduced. This criterion requires that a user who clicks according to a specific random click model should not create any preference in expectation. Using this criterion to guarantee absence of the biases, Radlinski & Craswell (2013) proposed to optimise interleaving sensitivity by increasing the uncertainty of the winner in a single interaction.

The core idea behind Probabilistic Interleaving (Hofmann *et al.*, 2011) is to represent document rankings as distributions over documents, with higher ranked documents having higher probability. When building an interleaved result page, documents are sampled according to these distributions. After the clicks are observed, the interleaving outcome is calculated by marginalising over all possible sampling paths that lead to the same result list. This marginalised estimator increases interleaving sensitivity. One of the specifics of Probabilistic Interleaving is that due to the random sampling of the results it allows combined ranking lists to have top results different from the top results of the compared systems. Indeed, even if both A and B have the result {https://www.facebook.com/} as the top result for the query "facebook", in the interleaved result page it can be placed on any position, even on the 10th, with a non-zero probability.

Below we describe the Team Draft interleaving algorithm, which we use or build upon throughout this thesis.[16] In Team Draft, the process used to generate the interleaved ranked list from two ranked lists mimics the process of splitting a set of players in two teams in a friendly football match. Firstly, two captains are selected. After that, they select players for their teams in turns. In Team Draft, each document corresponds to a player, and the rankings A and B represent the preference orders of the captains. At each turn, a coin is tossed to select the captain who selects the players next. This captain then selects their most preferred player among the players who were not included in the result list so far, and adds this player to the interleaved result list. These turns are repeated until the entire interleaved result page is built.

A formal description of the Team Draft interleaving algorithm is provided in Algorithm 2.2. In Figure 2.1 we illustrate possible results of applying Algorithm 2.2 on result pages with four results. Throughout this work, we refer to the set of the generated interleaved result pages as $\mathbb{L}$. Since after each

[16] In most cases, except for Chapters 6 and 8, other interleaving algorithm can be used instead. However, we use Team Draft due to the availability of a large dataset of Team Draft-based experiments, as this allows us to re-use real-world data in our experiments.

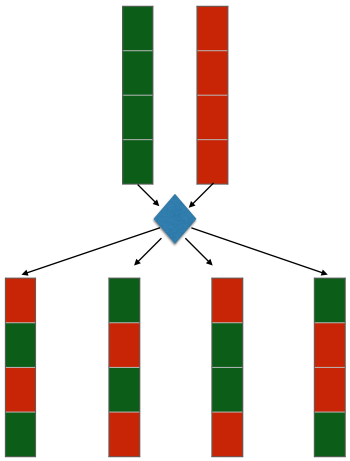


Figure 2.1: An illustration of the Team Draft interleaving algorithm on an example where result pages contain four documents. Red and green bricks correspond to documents that are associated with "teams" A and B, respectively.

coin toss in Algorithm 2.2 two documents are added to the interleaved list, $n/2$ coin tosses are needed to generate an interleaved result page with n results. Consequently, if we are working with four results per result page, there are $2^{4/2} = 4$ possible team assignments or combinations on the first result page (each combination corresponds to a "colouring" in Figure 2.1). We also enumerate these combinations as L_i, $\mathbb{L} = \{L_i | i = 1, ...\}$. In the case of ten documents per result page, there are 32 possible team combinations on the first result page, which can be enumerated as follows: $L_1 \leftarrow ababababab$, $L_2 \leftarrow abababababba$, ... , $L_{32} \leftarrow bababababa$.

From Algorithm 2.2 it is clear that each possible team combination L_i is generated with equal probability $\frac{1}{|\mathbb{L}|}$. Further, the probability distribution used to define how often a particular result page is shown to a user is called the interleaving *policy*. Throughout this thesis we denote the vector of probabilities that specifies the interleaving policy as $\boldsymbol{\pi}$, i.e. π_i is the probability of showing the result page L_i. While Team Draft uses a uniform policy, other algorithms, such as Optimised Interleaving (Radlinski & Craswell, 2013) and our proposed Generalised Team Draft (Chapter 6), use non-uniform probabilities.

To proceed, we define an *interaction* to be the part of a user session that starts when a user submits a query and ends either when the user submits another query or ends their session and leaves the search engine. In interleaving, scores of the alternatives are calculated by aggregating scores over all interactions in the experiment.

Input: Rankings $A = (a_1, a_2, ...)$ and $B = (b_1, b_2, ...)$
Output: Interleaved ranking list L, $TeamA$, $TeamB$
The interleaved ranking
$L \leftarrow ()$
while $(\exists i : A[i] \notin L) \wedge (\exists j : B[j] \notin L)$ **do**
  if $(|TeamA| < |TeamB|) \vee (|TeamA| = |TeamB| \wedge (RandomBit = 1))$ **then**
    Top result in A that is not included in L
    $k \leftarrow \min_i\{i : A[i] \notin L\}$
    $L \leftarrow L + A[k]$
    Clicks on the added result will be credited to A
    $TeamA \leftarrow TeamA \cup \{A[k]\}$
  else
    Top result in B that is not included in L
    $k \leftarrow \min_i\{i : B[i] \notin L\}$
    $L \leftarrow L + B[k]$
    Clicks on the added result will be credited to B
    $TeamB \leftarrow TeamB \cup \{B[k]\}$
  end
end

Algorithm 2.2: Team Draft algorithm used to interleave two result lists, as proposed by Radlinski *et al.* (2008)

After an interleaved result page is generated, it is shown to the user. To derive what alternative (A or B) the user preferred in an interaction, their clicks are analysed. How to interpret the user feedback in an interleaving experiment is an active area of research, which we discuss in more details in Chapter 6. In the simplest case, the analysis can be performed as follows. Firstly, each click in an interaction is associated with a team (A or B) that contributed the document on the clicked position[17]. To get the result of the experiments, the relative number of clicks is compared for both alternatives.

To formalise the analysis of interleaving experiments, we introduce the notation used in this thesis. Firstly, we fix an interleaving scoring scheme S. This scoring scheme associates a score $S(a)$ with an interaction a. We assume that a positive score indicates that the alternative system B is preferred in this interaction, and a negative score indicates that A is preferred. After running an experiment, the mean score statistic Δ is calculated:

$$\Delta = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} S(a) \tag{2.6}$$

where $\mathbb{A}$ is a set of the user interactions in the experiment.

A variety of interleaving scoring schemes were considered in the literature, most of them are heuristics (Chapelle *et al.*, 2012), but a machine-learned scheme was also proposed (Yue *et al.*, 2010). We discuss them in more detail in Chapter 6. However, since interleaving experiments are also used in

[17] Note, that even each document can be included in the interleaved list only by one of the teams.

Chapters 5, 7 & 8, we briefly review a widely used scoring scheme. This scheme is referred to as the *deduped binary* scheme, and it was proposed by Radlinski & Craswell (2010) and modified in (Chapelle *et al.*, 2012). This name reflects the fact that this scheme both accounts for the duplicate results in A and B, and has binary per-interaction outcomes.

Under the deduped binary click scoring scheme, in each interaction the team (A or B) which got more clicks is considered as a winner. A winner is assigned with a unit credit (i.e. 1). Importantly, if both rankings A and B have identical top-k results, clicks on these results are ignored. Indeed, if top-k results are identical, then after running Algorithm 2.2 the same results would form the first k results in the interleaved result list. However, the teams of these results would vary depending on the results of the coin tosses in Algorithm 2.2. Effectively, clicks on these results would be randomly assigned either to A or to B. By assigning these clicks with zero credit, we integrate this zero-mean noise out. Finally, interactions where both A and B obtained equal amounts of clicks are considered as ties, and they are broken uniformly at random between A and B.

It is convenient to calculate the outcome of an experiment with the binary deduped scheme used as suggested by Chapelle *et al.* (2012):

$$\Delta = \frac{winsB + \frac{1}{2}ties}{winsA + ties + winsB} - \frac{1}{2} \tag{2.7}$$

As in Equation (2.6), positive values of Δ in Equation (2.7) indicate that B outperforms A. How to infer if these improvements are statistically significant, is discussed in the next section.

2.3.3 Statistical Testing in Online Evaluation

As a result of an online experiment, a dataset of observations is collected. The goal of the statistical significance analysis is to infer if the difference observed between two systems in the experiment is due to a random chance or if indeed it indicates the preference of the users.

Usually, the analysis is performed by comparing two mutually exclusive hypotheses. Informally, the first hypothesis, referred to as the *null hypothesis*, states that there is no real difference between the compared systems. Under the *alternative* hypothesis, the systems A and B differ. The formal definitions of the tested hypotheses depend on the online evaluation method used and we discuss them below.

Before an experiment is started, the experimentation team decides on the acceptable level of statistical significance, α. The typical values of α used are 0.05, 0.01, and 0.001. These values correspond to 95%, 99%, and 99.9% confidence levels. Once the experiment is finished, the *p-value* statistic is calculated using the obtained dataset. The *p-value* indicates the probability of obtaining the difference

between systems at least as extreme as the one observed in the experiment assuming that the null hypothesis H_0 holds (i.e., if the systems are equally likely to satisfy the users). After that, the obtained *p-value* is compared to the significance level α. If *p-value* is smaller than α, then the null hypothesis H_0 is rejected and it is concluded that the tested systems differ. Otherwise, it is concluded that based on the experiment data it is impossible to reject H_0 on the selected significance level.

From the above described procedure it is clear that the probability of rejecting the null hypothesis when it holds is equal to α. Such an error is referred to as a *Type I error*. The opposite case — of rejecting the alternative hypothesis when it holds — is denoted as a *Type II error*.

To formalise the compared hypotheses in the case of A/B testing, we assume that we are interested in a fixed absolute online metric. Denoting the expected values of this metric of interest in the treatment and control groups as μ_B and μ_A, the goal of the statistical significance analysis is to infer if these means are different. Under the null hypothesis H_0, the expected values of the metric on both groups of users are equal:

$$H_0 : \mu_A = \mu_B$$

Under the alternative hypothesis H_1, these means are different:

$$H_1 : \mu_A \neq \mu_B$$

In the case of interleaving-based online experiments, the compared hypotheses are formulated similarly, with the absolute online metrics replaced by a relative online metric. An important difference with A/B testing is that in interleaving there is only one sample of observations, as A and B are mixed within each interaction. This allows interleaving to reduce between-user noise and, as a result, to achieve a higher convergence rate (Schuth *et al.*, 2015). Formally, the test is performed to compare the interleaving mean score Δ (Equation (2.6)) to zero (A and B are getting equal amounts of credits, but are assigned with different signs):

$$H_0 : \Delta = 0$$

The alternative hypothesis assumes that the mean score is not zero:

$$H_1 : \Delta \neq 0$$

Sometimes, a binary scoring scheme is used, such as the deduped binary scheme (Section 2.3.2) which assigns a binary score to each interaction, corresponding to the winning alternative (e.g. $+1$ if B wins and -1 if A wins). Denoting by p_B (p_A) the probability of A (B) winning in an interaction, and assuming that ties are broken between A and B uniformly, the hypotheses can be formulated as follows:

$$H_0 : p_A = p_B = 0.5 \quad H_1 : p_A \neq p_B$$

As a result, statistical tests that assume binary input can be used, such as binary sign test (Chapelle *et al.*, 2012).

2.4 Comparing Offline and Online Evaluation Approaches

After discussing offline and online evaluation approaches in Section 2.2 and 2.3, respectively, in the following we briefly overview their specifics in the form of a mutual comparison along several dimensions. This allows the highlighting of the strong and weak aspects of these two approaches.

Data re-usability In the offline evaluation approach, once a test collection is built, it can be repeatedly used to evaluate effectiveness of ranking of new systems. It is possible that the judgements or documents become outdated with time, hence a constant labelling process can be used to keep them up-to-date. In contrast, the data obtained in online experiments cannot be re-used to substitute new online experiments.[18]

Reproducibility Given exactly the same ranker and the same test collection, offline evaluation produces the same result. In online experimentation, several factors affect the reproducibility of the same comparison. Firstly, the user's needs evolve and change, e.g. according to the day of the week and season. Secondly, in online evaluation, systems are not separated from the remaining world when comparison if performed. For instance, during a typical two-week A/B test, several search engine's sub-systems can change, so that the same experiment repeated a week later theoretically might have a different outcome.

Effects on users During offline evaluation experiments, no users are involved, so their search experience cannot be harmed. However, when a change that negatively affects the user search experience is evaluated online, it can frustrate the users. Clearly, if a ranker with low effectiveness is evaluated in an A/B test, the users in the treatment group of the test would get rankings served by this bad ranker. It is possible that these users will not use this search engine again. This effect is less severe in the case of interleaving, as the users obtain the result pages that always contain results from both rankers.

Evaluation efficiency and scalability Assuming that a test collection is available, in system-based evaluation the experiment time is essentially bounded only by the search engine's run-time efficiency, thus the experiments are fast to be performed. In contrast, online experiments tend to last for days and weeks. A typical A/B test is deployed for a time period that lasts for weeks (Chapelle *et al.*, 2012; Drutsa *et al.*, 2015; Kohavi *et al.*, 2013). Interleaving experiments tend to be shorter, typically lasting

[18] Some studies (Grotov *et al.*, 2015; Li *et al.*, 2015) aim to re-use historical interaction data so that new online experiments are not required, but they are limited in their applicability (e.g. they cannot be used to evaluate changes in UI, or ranking with new documents) and cannot be considered as online experiments.

for several days, e.g. Chapelle *et al.* (2012) used interleaving experiments from Bing and Yahoo! that are from two to five days long. We consider low evaluation efficiency to be one of the major problems of the online evaluation, and a part of this thesis (Chapters 6, 7, and 8) is devoted to addressing it.

In addition, online evaluation is also limited by the bounded resources it uses. Indeed, in a shared-control scenario, each A/B experiment is deployed for two weeks and takes up to 10% of the search engine's users as the treatment group (Kohavi *et al.*, 2013) and 10% of the user population is used as the shared control group. Hence, in such a configuration it is only possible to run up to 9 experiments affecting ranking of the results in two weeks.[19] Clearly, these restrictions limit the usefulness of online evaluation. Apart from increasing evaluation efficiency, one can try to build a system that selects what online experiments are worth deploying first. We discuss this approach in Chapter 5.

Evaluation effectiveness The state-of-the-art offline metrics used in system-based evaluation are designed to model user's satisfaction with the evaluated ranking. While some of these models are elaborate (Chapelle *et al.*, 2009; Chuklin, Serdyukov & de Rijke, 2013; Yilmaz *et al.*, 2010), they only tend to approximate the user behaviour. In online evaluation experiments, and in particular in A/B tests, we directly observe how user behaviour changes after the system is changed. In that sense, online experimentation has a higher potential than offline evaluation methods. However, the question of how to choose an online evaluation metric such that it reliably represents the user's satisfaction with the search engine and is sensitive is an open problem (Section 2.3.1).

Applicability While system-based evaluation methods and interleaving can only be used to evaluate the ranking effectiveness, A/B tests have a very broad area of applicability: an A/B test can be used to assess the user satisfaction with a user interface (UI) change, the effectiveness of document ranking, snippet or QAC quality. Further, so far interleaving evaluation was only reported to be used in the document search domain. We hypothesise that it also can be applied to other domains with list-based representation of the results, such as QAC or snippet evaluation, however, to the best of our knowledge there are no studies published that support the validity of this hypothesis. In Chapter 6 we demonstrate that interleaving methods can be extended to the domains with grid-based result representation, such as image search.

Cost of operation The cost of building a large-scale test collection for offline evaluation can be substantial, as it requires obtaining relevance judgements, which is a labour-intensive process performed by humans. On other hand, once the test collection is built, it can be repeatedly re-used. Online experiments can be deployed at no direct cost, assuming that the corresponding infrastructure is implemented

[19] As we discuss in Section 3.4, a single user might participate in several experiments simultaneously, if these experiments affect different "variables", such as ranking and UI.

and the market share is not lost due to testing low-quality changes.

Overall, from the comparison of the evaluation approaches in this section above and in Sections 2.2 and 2.3, we notice the both approaches are in fact very complimentary. For example, since offline evaluation cannot harm the search engine's users and can be performed fast and at no cost (provided the test collection is available), we can use it extensively to filter changes that can damage the experience of the search engine's users. In other words, as running online evaluation is a long process potentially affecting the users, we generally need to use offline experimentation to avoid these possible downsides. In turn, as online experiments are the most direct way to predict the user's reaction to a change in a search engine, we might want to use it as often as possible; but as the online evaluation can harm the users and its throughput is limited, we need to use the offline methods to ensure high quality of the experiments that go to the online evaluation. All these observations motivate us to consider the *evaluation pipeline*, a combination of the offline end online evaluation steps that are used sequentially to ensure that a search engine makes fast and correct decisions when evaluating its improvements. In Chapter 3 we discuss the model of the evaluation pipeline used in this thesis.

Before proceeding to discuss the evaluation pipeline, in the next section we overview some of the important models of the user clicking behaviour. These models are tightly connected to offline effectiveness metrics (Chapelle *et al.*, 2009; Chuklin, Serdyukov & de Rijke, 2013; Yilmaz *et al.*, 2010) and are intertwined with our work in Chapters 4, 5, and 6.

2.5 Click Models

Since it was demonstrated that implicit user feedback can be effectively used for evaluating rankers (Joachims, 2002, 2003), the problem of better understanding of the user's clicking behaviour started to attract researchers. Joachims *et al.* (2005) observed that the user clicking behaviour is informative, but biased. The first type of bias Joachims *et al.* (2005) described is the *trust* bias: users trust their search engine's ranking ability and click on the first result even when it is less relevant than the second result. The second type of bias is the *quality* bias: if the quality of the ranking decreases, the users start to click on less relevant results. This indicates that the clicking behaviour is dependent on the overall quality of the results. Hence, the implicit feedback can only provide relative preference information. Joachims *et al.* (2005) also proposed several strategies to extract pairwise preferences from the user clicks, i.e. a clicked document is likely to be more relevant than documents ranked higher but not clicked.

Later, a variety of user models were proposed that aim to account for these biases in the user feedback data (Chapelle & Zhang, 2009; Craswell *et al.*, 2008; Dupret & Piwowarski, 2008; Dupret *et al.*,

2007; Guo *et al.*, 2009; Moffat & Zobel, 2008). An overview of some of the most representative click models can be found in (Chuklin *et al.*, 2015). Below we review two models that are most relevant to our work, the cascade click model and the Dynamic Bayesian Network model.

The cascade model (Craswell *et al.*, 2008) assumes that the user examines results from top to bottom, sequentially. The top result is always examined. At each step, the user decides whether to move to the next result or to click on the current. Once the user clicked, they never return to the result page; the user continues examination until a result is clicked.

To formalise the model, we assume that a query and a result page is fixed, and introduce the following binary random variables. E_u indicates if the document u was examined, C_u denotes if u was clicked, and A_u denotes an event of the user finding the document u attractive. By u_i we refer to the document ranked on the ith position. Under this notation, the cascade model is described by the following Equations (2.8a)-(2.8f) (Chuklin *et al.*, 2015):

$$C_u = 1 \Leftrightarrow E_u = 1 \ \ and \ \ A_u = 1 \tag{2.8a}$$

$$P(A_u) = \alpha_u \tag{2.8b}$$

$$P(E_{u_1} = 1) = 1 \tag{2.8c}$$

$$P(E_{u_i} = 1 | E_{u_{i-1}} = 0) = 0 \tag{2.8d}$$

$$P(E_{u_i} = 1 | C_{u_{i-1}} = 1) = 0 \tag{2.8e}$$

$$P(E_{u_i} = 1 | C_{u_{i-1}} = 0, E_{u_{i-1}} = 1) = 1 \tag{2.8f}$$

Equations (2.8a)-(2.8f) state that: (2.8a) the user clicks on a results if and only if the results is examined and attractive; (2.8b) the probability of attracting the user is the document's parameter α_u; (2.8c) the top result is always examined; (2.8d) the results are examined sequentially from top to bottom; (2.8e) the user stops examination after a click; (2.8f) the results are examined until the user clicks.

This model is relatively basic (e.g. it cannot model multi-click interactions), but it forms a foundation for the ERR effectiveness metric (Chapelle *et al.*, 2009) and provides a foundation to a number of other click models, including the model of the user interactions with query auto-completion mechanisms we propose in Chapter 4.

Dynamic Bayesian network model (DBN) proposed by Chapelle & Zhang (2009) is also based on the cascade model. In contrast to the cascade model, under DBN the user might click several times, and stops either if the clicked document satisfies them or if the user decides to abandon the search. To model that, Chapelle & Zhang (2009) introduced another document parameter, s_u, that specifies the

probability of satisfying the user after click. In addition, they introduce a constant probability $(1-\gamma)$ of user deciding to abandon the interaction. The DBN model is specified by the following Equations (2.9a)-(2.9h):

$$C_u = 1 \Leftrightarrow E_u = 1 \ \ and \ \ A_u = 1 \tag{2.9a}$$

$$P(A_u) = \alpha_u \tag{2.9b}$$

$$P(E_{u_1} = 1) = 1 \tag{2.9c}$$

$$P(E_{u_i} = 1 | E_{u_{i-1}} = 0) = 0 \tag{2.9d}$$

$$P(S_{u_i} = 1 | C_{u_{i-1}} = 1) = s_u \tag{2.9e}$$

$$C_u = 0 \Rightarrow S_u = 0 \tag{2.9f}$$

$$P(E_{u_i} = 1 | S_{u_{i-1}} = 1) = 0 \tag{2.9g}$$

$$P(E_{u_i} = 1 | E_{u_{i-1}} = 1, S_{u_{i-1}} = 0) = \gamma \tag{2.9h}$$

Equations (2.9a)-(2.9h) closely resemble the specification of the cascade model in Equations (2.8a)-(2.8f) with the following difference: (2.9e) after a click on the document u the user is satisfied with the document-dependent probability s_u; (2.9f) only clicked documents can satisfy the user; (2.9g) a satisfied user stops examination; (2.9h) if not satisfied, user continues examination with probability γ. The query-document parameters are estimated by an EM-based procedure (Chapelle & Zhang, 2009).

An important modification of the DBN model is the simplified DBN model (sDBN). This model assumes that the user always continues to examine the results, i.e. $\gamma = 1$. In that case, the last clicked result satisfied the user. As a result, the model parameters can be estimated by a simpler procedure, however, sDBN has predictive capabilities comparable to that of DBN (Chapelle & Zhang, 2009). We provide the details of the parameter training procedure in Algorithm 2.3. We use this sDBN model in Chapters 5 and 6.

2.6 Conclusion

In this chapter, we provided a high-level overview of the evaluation techniques and related topics. We split the overview of the evaluation techniques in two parts, discussing offline evaluation in Section 2.2 and online evaluation in Section 2.3. The area of offline evaluation (also referred to as system-based evaluation), was actively developing since the 1960-s and has its roots in the Cranfield experiment (Cleverdon, 1967; Voorhees, 2002). In contrast to offline evaluation, online evaluation methods

Input: Set of user interactions $\mathbb{A}$, Beta priors on the click model parameters: $\alpha_a, \alpha_s, \beta_a, \beta_s$
Output: The click model parameters for each document u: a_u, s_u
$a_u^N \leftarrow 0$; $a_u^D \leftarrow 0$
$s_u^N \leftarrow 0$; $s_u^D \leftarrow 0$
foreach *interaction in* $\mathbb{A}$ **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 the current interaction
  $s_u^N \leftarrow s_u^N + 1$
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 2.3: Training the sDBN model, as described in (Chapelle & Zhang, 2009). Following Chapelle & Zhang (2009), we use uniform Beta priors $\alpha_a = \alpha_s = \beta_a = \beta_s = 1$ throughout this thesis.

attracted interest of the IR research community relatively recently, probably due to being increasingly popular in the industrial setting.

Both offline and online evaluation approaches have their strong and weak sides. Indeed, offline evaluation comparisons are easily reproducible and, given a test collection, are fast to be obtained. However, there is a spectrum of possible changes (e.g. UI changes or personalised search) that are either hard or impossible to evaluate offline at scale. In contrast, online experiments tend to be time-consuming and require a part of the limited resource of user traffic, but they better reflect the needs of the users than offline evaluation experiments.

As we discussed in Section 2.4, the offline and online evaluation approaches can be used in conjunction, so that the overall evaluation process leverages strong sides of the both approaches to guarantee a fast and progressive development of a search engine. Combined, they form an evaluation pipeline that we discuss in the next chapter.

In Section 2.5, we provided a focused review of the click models we use in this thesis, namely the cascade click model and the Dynamic Bayesian Network click model. These models form a foundation to some of our further work in Chapters 4, 5, and 6.

Before going into detailed discussion of the evaluation pipeline, we want to highlight that instead of concentrating on a single part of this pipeline, in this thesis we aim to improve the evaluation pipeline as a whole by improving its individual parts. We discuss the roadmap for achieving this in Section 3.5.

Chapter 3

Evaluation pipeline

3.1 Introduction

As we discussed in Section 1.2, data-centric evaluation is an important tool that ensures progressive development of search engines. We denoted by *evaluation pipeline* a combination of steps that are used to assess the quality of the deployed changes. The evaluation pipeline specifies a typical life cycle of a change from the moment it was developed to the moment a decision is made if the change is useful or not. Further, either this change is accepted as *successful* and becomes a part of the production system or is rejected, if it is proven not to be useful. Typically, web search evaluation is a multi-stage process.

An important requirement for the evaluation pipeline as a decision mechanism is its *accuracy*. In other words, we require that changes that degrade the users' search experience should be accepted as rarely as possible. Another important requirement for the evaluation pipeline is *efficiency*. We define efficiency as the ability to produce as many successful changes as possible in a unit of time. If, given a fixed stream of changes, an evaluation pipeline produces twenty successful experiments in a week, we consider it to be more efficient than one that produces only ten. A higher efficiency can be achieved either by processing experiments faster, or by early rejecting experiments that are unlikely to be successful. In this thesis, we investigate both possibilities.

As discussed later in Section 3.3, in a possible evaluation pipeline implementation, the majority of the changes deployed by a search engine are often assessed not to harm the search engine's users by means of the rigorous, multi-stage testing. This implies that the efficiency of the evaluation pipeline as a whole is one of the limiting factors of the search engine's progress rate. Indeed, if evaluating a single hypothesis takes more than two weeks, how long it will take to iterate over several unsuccessful hypotheses before a successful one is found? The successful to unsuccessful experiments ratio was reported to be relatively low in modern search engines, e.g. in 2009 in Bing two in three hypotheses tested

in online experiments proved to be unsuccessful (Kohavi *et al.*, 2009) and were rejected. Furthermore, as search engines become increasingly advanced, improving them becomes harder and thus the ratio is likely to decrease further in time.

To illustrate the evaluation pipeline, we start this chapter by Section 3.2, where we go through a possible evaluation process for three example changes in the search engine. In the first change, a new ranking feature is added to the ranking algorithm. In the second example, we follow the evaluation of a change in the ranking of query auto-completions. Finally, in the third example, we evaluate a change in the user interface. Based on these illustrative examples, in Section 3.3 we discuss a possible implementation of the evaluation pipeline in a search engine. For the purposes of this thesis, a high-level description of the pipeline is sufficient. In general, parts of the pipeline we discuss below can be skipped in some cases (e.g. the offline evaluation step might be skipped in the evaluation of a change in the user interface) or altered. Moreover, the steps themselves might be more sophisticated than we describe in this chapter. For instance, in Section 3.4 we go a bit deeper in describing how elaborated an online evaluation step can be.

However, the basic picture of the evaluation pipeline we describe in this chapter is sufficient for the purposes of this work, and we describe all parts of the pipeline that are relevant to this thesis. Based on this description, in Section 3.5 we discuss a roadmap for improving the efficiency of the evaluation pipeline that we follow in this thesis. At the same time, while improving the evaluation pipeline's efficiency, we aim to maintain its accuracy level.

3.2 Three Evaluation Examples

We start by describing a possible evaluation scenario for three typical types of changes in a search engine: a change in the learning to rank algorithm for the document search, a change in the ranking of query auto-completions, and a change in the user interface.

Ranking algorithm Assume, we want to assess a hypothesis that states that adding a new ranking feature in the ranker increases the quality of the ranking. Typically, this hypothesis would be assessed as follows.

Firstly, the ranking algorithm, e.g. a LambdaMART (Burges, 2010) implementation, is trained twice on a collection of the query-document pairs that are manually labelled by relevance judges: with and without the tested ranking feature. This training is aimed to optimise a specific offline evaluation metric, such as nDCG (Järvelin & Kekäläinen, 2002) or ERR (Chapelle *et al.*, 2009). The training is performed in a cross-validation setting. After the training is finished, we test a statistical hypothesis that assumes

that the ranking algorithm with a new ranking feature has higher mean of the metric of interested. If the mean value of the metric is indeed increased, and the improvement is accepted as sufficiently large and statistically significant, the hypothesis proceeds to the next step.

In the next step, the trained model is deployed to a copy of the production system and is evaluated offline. However, in contrast to the earlier evaluation, this time the change is evaluated in conjunction with the remaining systems that comprise the search engine. Importantly, some of them might affect the ranking of the results (e.g. a re-ranking system that promotes fresh results). After that, a hold-out offline evaluation collection is used to assess the quality of the ranking. Once a statistically significant improvement is achieved, the evaluation of the hypothesis proceeds to the next step, namely the online experiments scheduling step.

Essentially, the above described steps are performed offline, in the system-based setup (Section 2.2), without any need for real users. If the tested change improves, or, at least, does not degrade the offline metrics, it is added to the online experimentation queue. In the case of the ranker change, the online evaluation can be performed either by interleaving or by an A/B test. It might be the case that an online experimentation slot (discussed in Section 3.4) is available so that the experiment is deployed immediately to evaluate the ranker with the added feature. It is also possible that there are no available slots, so the experiments are scheduled with some priority to be executed later.

After the online experiment is deployed for a fixed and pre-defined time period, e.g. a week, it is terminated and a statistical analysis is performed to infer if there is a statistically significant difference between the compared systems: if so, the new ranking feature is added to the ranking system.

QAC change The evaluation of a change in the ranking of query auto-completions (QAC) follows a path similar to the one of a change in document ranking. Firstly, it is evaluated offline, against a set of queries submitted by the search engine's users earlier. If the quality of the ranking as measured by a chosen offline evaluation metric is increased, the tested change proceeds to the online evaluation step. The result representation typically used in QAC is similar to the representation used in document search and, theoretically, interleaving can be used to compare the tested change to the QAC ranking used in production. However, to the best of our knowledge, no studies validate or use interleaving in the QAC domain. Thus, we assume that an A/B test is performed to evaluate the proposed change. Similar to the previous example, the online evaluation experiment is either deployed immediately or is scheduled to be deployed later. After the corresponding online experiment is finished, a statistical analysis test is performed to infer if the difference between the absolute online metric[20] of the proposed change and the current system is statistically significant.

[20] Such a metric can measure, for instance, how often the QAC mechanism is used by the users.

Importantly, in our document ranking and QAC examples, the chances of rejecting a tested change in online experiments are higher, if the offline evaluation metrics used to tune and test the changes are not aligned with the online metrics. How we plan to address that is discussed in Section 3.5.

UI change When evaluating a change in the user interface (UI), we are not capable of using the system-based evaluation, as in such an evaluation we ignore the actual representation of the results. As a result, the offline evaluation is either not performed at all, or is performed as a small-scale study by asking a group of participants to compare two possible UI implementations side-by-side (Thomas & Hawking, 2006). However, the online evaluation is still useful. Notably, an interleaving evaluation experiment is not applicable here, as it is designed for the comparing two alternative rankings. As a result, the evaluation can only be performed by an A/B test. In such a test, the current used UI is compared with the changed UI. To test the hypothesis that the system with the changed user interface better satisfies the users, a corresponding A/B experiment is added to the online experimentation queue. If there is an available slot in the corresponding experimentation layer, the experiment is deployed. Similarly to the earlier ranking evaluation example, the experiment is performed for a fixed time period, usually for a week or two. After the experiment is stopped, the collected data is used to analyse if there is a statistically significant difference in the user satisfaction with the compared systems.

3.3 Pipeline Structure

Based on the examples discussed in Section 3.2, we highlight the steps that a possible improvement in a search engine passes by before it is accepted as a part of the search engine or it is rejected. These steps essentially correspond to the parts of the evaluation pipeline which are discussed in Section 3.3.2.

3.3.1 An Improvement's Life Cycle

Offline optimisation Often, a changed subsystem in the search engine has numerous parameters that are optimised in a dedicated specific machine learning step. A classic example is a search engine's ranking algorithm, which is learned to optimise an offline metric on a labelled dataset. This process precedes the evaluation process we concentrate on in this thesis.

Offline evaluation Once the parameters of the changed subsystem are optimised, its quality is evaluated against a test dataset. For instance, in the case of a change in the ranking algorithm, its effectiveness is assessed on a separate bucket of labelled queries. This *offline evaluation step* might differ from the evaluation in the offline machine learning process as it evaluates the search engine as a whole. In our ranking evaluation example, a new ranking algorithm is evaluated in conjunction with other parts of the search engine that affect ranking, such as verticals.

Online experiment scheduling Once the offline evaluation demonstrates that the tested change does not degrade the offline quality metric, it proceeds to the online evaluation step. However, as each online experiment consumes a part of the query traffic and might last for a considerable period of time (e.g. a week), sometimes it can be impossible to deploy all the tested changes for online evaluation simultaneously. As a result, a queue of the experiments is organised. In an extreme case, the experiments might come to the queue faster than performed. As a result, the problem of selecting the most promising experiments for future evaluation arises. For this reason, we consider a separate *online evaluation scheduling step* where such a selection is performed.

Online evaluation In the *online evaluation step*, the quality of the tested change is assessed by exposing the real users of the search engine to the tested change. This is performed usually by means of an A/B test, or in an interleaving experiment, as discussed in Section 2.3. A part of the online evaluation step is the statistical analysis of the obtained results (Section 2.3.3).

Deployment After assessing the quality of the tested change in an online experiment, it can be accepted as successful and considered for future deployment. Otherwise, it can be considered for a further improvement and re-evaluation. We consider this step not to be a part of the evaluation pipeline.

Reverse testing For a variety of reasons, one might wonder if an already deployed change (feature) is still useful. For example, due to changes in the user needs or due to other new improvements in the search engine, some features might become redundant. In that case, a reverse test can be used to assess if it is indeed redundant. Such a test can be performed, for instance, by running an A/B test where one of the alternatives is obtained by switching the deployed change off. In a sense, this step corresponds to the same online evaluation step where the change tested just "turns off" an earlier deployed change. Hence, we do not differentiate such a test from a regular evaluation experiment.

3.3.2 Structure of the Evaluation Pipeline

The structure of the evaluation pipeline we consider in this thesis reflects the life cycle of a change described in Section 3.3.1. In Figure 3.3.2 we provide its schematic representation. This simple model consists of three steps: offline evaluation, scheduling, and online evaluation steps. In the offline evaluation step the assessed change is tested to improve (or at least not to degrade) the offline evaluation metrics. After that, this change proceeds to the scheduling step where it is arranged in the experimentation queue according to how promising is this particular change. After the corresponding online experiment is deployed, the user behaviour data representing the user's satisfaction in the experiment is collected. After the experiment is finished, the collected data is used to test the statistical significance of the difference between the systems.

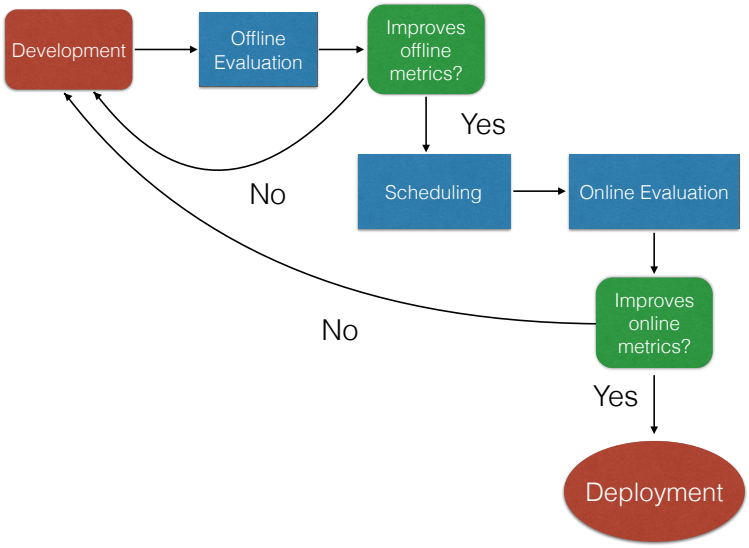


Figure 3.1: A schematic model of an evaluation pipeline considered in this thesis.

Despite being a high-level schematic description of a search engine's evaluation pipeline, this model highlights how the overall efficiency of the evaluation pipeline can be improved. In the following, we firstly discuss one of the existing approaches to increase the online evaluation scalability (Section 3.4). After that, in Section 3.5 we discuss the roadmap for improving the efficiency of the evaluation pipeline as a whole that we will follow in this thesis.

3.4 Overlapping Online Experiments

A straightforward way to improve the scalability and thus the efficiency of online evaluation is to allow each user interaction to participate in several online experiments at the same time. However, a special care is needed while designing such an evaluation system due to possible interferences between the experiments. In an example discussed in (Tang *et al.*, 2010), two user interface (UI) changes are considered. The first change alters the parameter that specifies the colours of the document links, and the second change modifies the parameter that controls the background colour of the result page. While blue colour is a possible value for the both parameters, setting them both to blue would make the search results unreadable and have catastrophic consequences to the users.

A possible solution is proposed by Tang *et al.* (2010) and its core idea is to organise online experiments in a layered, overlapping structure. In order to achieve this, the set of controlled parameters is

split in subsets, with subsets corresponding to experimentation layers. Each layer is used to run experiments that alter user experience along one dimension, e.g. it is possible to have ranking, user interface, query auto-completion, and snippet layers.

Such a system allows to use a single user interaction in several experiments, one per layer, while avoiding experiments influencing each other. A similar approach was reported to be used in Bing (Kohavi, 2012).

In a possible implementation, the users can be represented by a cookie, and, by means of adding a salted hashing of the cookies, they can be randomly split in 20 groups, for example. Each group contains 5% of users, and each A/B test experiment in a layer uses two groups of users, one for A and one for B. Assuming that the control group (i.e. A) can be shared across the experiments, in this example it is possible to run up to 19 experiments per layer simultaneously.

While the described multi-layer system increases the scalability of the evaluation pipeline, the problem of improving the evaluation efficiency remains. Indeed, in some layers of the evaluation more experiments can be required than evaluated. For instance, in the ranking level it is relatively cheap to devise an improvement by changing the parameters of the learning to rank algorithm. Moreover, the evolution rate of the search engine is still hindered by the efficiency of the individual experiments. As a result, the problem of achieving a higher efficiency in the single layer remains actual.

3.5 Roadmap for Improving the Evaluation Pipeline

How to improve the efficiency of the evaluation pipeline is the central question of this thesis. We hypothesise that *by using the historical interaction data, each step of the evaluation pipeline can be improved so that the overall efficiency of the evaluation pipeline is increased*. Specifically, we propose the following roadmap to improve each step of the pipeline so that the efficiency of the pipeline as a whole is increased:

1. To reduce the number of experiments that pass the offline evaluation step, but are rejected after the online evaluation step, we propose to use machine-learned offline metrics that are optimised to be better aligned with the online user preferences. This optimisation is performed on the historical user interaction data. As a result, a better effectiveness as indicated by the offline metric is more likely to imply a better effectiveness as measured in an online evaluation. Thus, less experiments pass the offline step to be later rejected during online evaluation. We follow this idea and propose offline evaluation framework for the query auto-completion domain in Chapter 4.

2. In Chapter 5, we propose a machine-learned online experiment scheduler that aims to deploy only the most promising experiments. In the situation when it is impossible to deploy all the scheduled experiments simultaneously, the optimised scheduler can increase the overall evaluation pipeline efficiency by reducing the number of unsuccessful experiments deployed for online evaluation, thus allowing a better utilisation of the limited resource of the user sessions;

3. As the online evaluation is the most time-consuming step, reducing its duration can notably increase the overall efficiency of the evaluation. We propose to achieve this reduction by both improving the sensitivity of online evaluation (Chapter 6) and by improving the way the data from online experiments is tested for statistical significance (Chapter 7). In Chapter 8, we argue that both ways to increasing online evaluation efficiency can be used in combination and lead to even higher gains in efficiency.

Overall, in this thesis we discuss how to improve each step of the evaluation pipeline so that the overall efficiency of the pipeline is improved. At the same time, we control, where possible, that the accuracy of the improved steps is not harmed.

3.6 Conclusions

In this chapter, we described a model of the typical evaluation pipeline used by commercial search engines. We started with Section 3.2, where we followed the possible evaluation steps for three different changes in search engines, namely a change in the document search ranking algorithm, a change in the ranking of query auto-completions, and an improvement in the user interface. Based on these examples, we presented a schematic model of the evaluation pipeline (Section 3.3). While this model is high-level and parts of it can be more sophisticated than we described, as demonstrated by Section 3.4, it is sufficient for the purposes of this work.

This model of the evaluation pipeline allowed us to introduce the roadmap for improving the evaluation process as a whole, and we discussed this roadmap in Section 3.5. According to this roadmap, we consequently work on improving each step of the evaluation pipeline. In particular, in Chapter 4, we work on improving the offline evaluation step. In Chapter 5, we improve the scheduling step. In Chapters 6, 7, and 8, we discuss how the efficiency of the online evaluation step can be improved. Overall, in each of these chapters, the historical user interaction data becomes an invaluable tool to improve the evaluation process as a whole, thus supporting the statement of this thesis.

Chapter 4

Framework for Offline Query Auto-completion Evaluation

4.1 Introduction

This chapter is based on a publication (Kharitonov *et al.*, 2013*b*) and studies the problem of offline evaluation of query auto-completion mechanisms. The query auto-completion mechanism (QAC) within a search engine is a tool aimed to help users to type less while submitting a query. In its most basic form, the list of suggested queries is formed by listing queries that start with the user's input as a prefix. A typical QAC interface is represented in Figure 4.1.

In this chapter, we discuss how the offline evaluation step can be improved by using the historical user interaction data. In particular, we investigate how the offline evaluation metrics can be improved in a typical web search application, namely query auto-completion (QAC). Despite QAC being used by all commercial search engines, QAC evaluation lacks a well founded offline evaluation metric that is aligned with the online user preferences. As we will demonstrate below in this chapter (Section 4.9), the previously existing metrics are only loosely aligned with online user behaviour. As a result, online evaluation experiments that test changes in QAC are more likely to be rejected during online evaluation. However, as discussed in Section 2.4, online experiments cost time and can annoy users if the tested change degrades the user experience. Hence, building an offine metric that is aligned with online evaluation is an important task and it is the gap we aim to address in this chapter. This direction of work corresponds to point one of the pipeline improving roadmap we outlines in Section 3.5.

A possible way to ensure the alignment is to design the metric on top of a realistic model of the user behaviour. For evaluation in the web search domain, Expected Reciprocal Rank (ERR) (Chapelle *et al.*, 2009) and Expected Browsing Utility (EBU) (Yilmaz *et al.*, 2010) are examples of user model-based

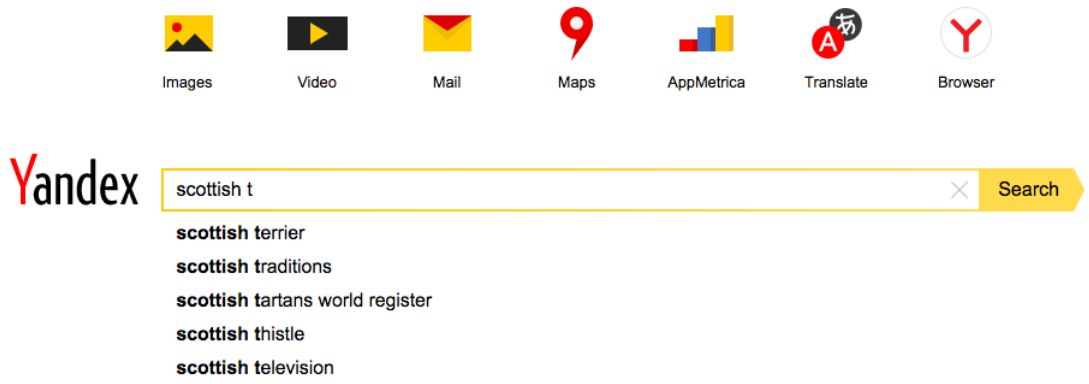


Figure 4.1: A search engine's interface with its query auto-completion mechanism suggesting a set of queries. The suggested queries contain the user input as their prefix.

effectiveness metrics. We follow a similar approach, by building a realistic model of the user behaviour and adjusting its parameters using historical interaction data to ensure a high alignment with the online behaviour of the users. This approach supports the statement of this thesis (Section 1.3): indeed, by re-using historical interaction data, we improve the efficiency of the evaluation pipeline as a whole.

In this chapter, we follow a similar approach. We propose a novel framework of the offline query auto-completion effectiveness metrics, called *Saved*. This metric framework is parameterised by a model of the user interaction with the query auto-completion mechanism and an effort function. The effort function characterises the level of effort it takes the user to submit their query. To instantiate our proposed framework, we consider several models of the user behaviour and two forms of the effort function.

Overall, we consider our contributions in this chapter to be as follows:

- We study the ways users interact with the query auto-completion mechanism, as it is observed in the session logs, and propose a cascade model of the user behaviour;
- We propose a family of effectiveness metrics, called *Saved*, parametrised by a user model and study several possible instantiations of these metrics;
- We perform a thorough experimental study of the considered user models and evaluation metrics in comparison to the evaluation metrics used in the literature.

The remainder of this chapter is organised as follows. After reviewing some related work in Section 4.2, we study how users interact with a query auto-completion mechanism and propose a user model in Section 4.3. In Section 4.4, we introduce an algorithm to learn the parameters of the user model from

the session log. Next, we briefly discuss the considered evaluation framework in Section 4.5. Further, we introduce a novel family of evaluation metrics, called *Saved*, and discuss their connection to other metrics used in Information Retrieval in Section 4.6. Section 4.7 describes a methodology we use to compare the proposed user models and metrics. The dataset used in our experiments is presented in Section 4.8. We report and discuss the experimental results in Section 4.9. We provide concluding remarks in Section 4.10.

4.2 Related Work

Our work in this chapter is based on the earlier research in the following areas: the user model-based evaluation metrics, the methods used to compare the effectiveness metrics, the models of user interaction with query auto-completion mechanisms, and the metrics used to evaluate query auto-completion mechanisms.

Models of the user search behaviour and their connection to the evaluation metrics gained a lot of attention in the document search domain and inspired us to follow this direction in our work in the query auto-completion evaluation. Thus, it is important to review the user model-based evaluation approach (Section 4.2.1) and the models of the user behaviour considered in the QAC domain (Section 4.2.2).

Once a new metric is developed, the question arises about how to compare it to the variety of existing metrics. This problem has also received some attention from the research community. We face the same problem of assessing the quality of the query auto-completion metrics. Thus we review how metrics evaluation is performed in Section 4.2.3. We finish the overview of the related work with the discussion of the methods and metrics used in the evaluation of query auto-completions previously used in the literature (Section 4.2.4).

4.2.1 User Model-inspired IR Metrics

One of the state-of-the-art web search evaluation metrics, Expected Reciprocal Rank (ERR), has a strong relation with the cascade model of the user behaviour and is defined as part of the cascade-based family of metrics (Chapelle *et al.*, 2009). We discussed the cascade model in Section 2.5. This model assumes that a user examines a list of ranked documents one-by-one, from top to bottom. After examining the document in the ith position, either the user is satisfied and stops the examination process or continues to the document in position $i + 1$. The probability of satisfying the user depends on the document's relevance. A cascade-based metric is defined as the expectation of a function $\phi(r)$, where r is the rank where the user finds the document they were looking for. In case of ERR, the function $\phi(r)$ is specified to be $\frac{1}{r}$.

An extension of ERR based on the cascade model with abandonment (Chapelle & Zhang, 2009) was also discussed by Chapelle *et al.* (2009). Apart from being based on a different user model, this extension leverages a different instantiation of the function $\phi(r)$, which is equal to 1 if the user finds a satisfactory result, and 0 otherwise. As a result, the value of this metric is equal to the probability of the user finding a relevant result as predicted by the underlying user model.

The Expected Browsing Utility (EBU) is another search effectiveness evaluation metric proposed by Yilmaz *et al.* (2010), which is defined as the expected document utility a user "collects" while examining a result list. At its basis, EBU uses a more sophisticated cascade user model that accounts for snippet attractiveness.

The effectiveness metrics that we introduce resemble the cascade family of the web search effectiveness metrics (Chapelle *et al.*, 2009), but applied to the query auto-completion domain with different user behaviour patterns.

A more recent work on user model-based evaluation metrics is (Chuklin, Serdyukov & de Rijke, 2013). In their work, Chuklin, Serdyukov & de Rijke (2013) classified the click model-based metrics as effort- and utility-based metrics. The utility-based metrics can be represented as the expectation of the utility of the clicked results under a specific click model. In contrast, the effort-based metrics represent the expectation of an effort a user has to put before finding a satisfactory document. Based on our proposed evaluation framework, we instantiate two metrics, *pSaved* and *eSaved*, that represent the probability of satisfying a user and the expected relative number of characters a user can skip while submitting their query. These two metrics can be considered as effort-based metrics in the definitions suggested by Chuklin, Serdyukov & de Rijke (2013).

4.2.2 Modelling User Interaction with QAC

The first model of user interactions with query auto-completion mechanisms was proposed in our work (Kharitonov *et al.*, 2013*b*) and we discuss it in detail in Section 4.3. Under this model, the user types their query in steps, character after character. After submitting a character, the user has a chance to examine the list of the queries that QAC suggests. Each position is examined with some probability that is a function of the position's rank and the length of the already typed prefix. The interaction ends either when the user selects a query from the list of suggested auto-completions or when the user finishes typing their query.

In a later work, Mitra *et al.* (2014) investigated what interaction parameters might affect the user's decision to use the query auto-completion mechanism. In particular, they noticed that the probability

of a user clicking on a query generated by QAC strongly depends on the rank of the suggested auto-completion. Furthermore, some features such as the distance from the end of the word, the fraction of the query typed, keyboard distance between the characters of the word also proved to be useful in predicting the user's attention.

In the first eye-tracking study of user interactions with QAC, Hofmann *et al.* (2014) observed that the users tend to examine the top-ranked queries even when their effectiveness is not different from the queries ranked lower. Thus, Hofmann *et al.* (2014) had confirmed the presence of a position bias in user interactions with the query auto-completion mechanisms.

Later, a click model for query auto-completion interactions was proposed by Li *et al.* (2014). This model differs from the model we consider in this chapter (Section 4.3) in several aspects. Firstly, it assumes that the user examines the list of the suggested completions from top to bottom, i.e. all queries ranked higher than an examined query are also examined. Next, the model proposed by Li *et al.* (2014) explicitly models the "relevance" of the query auto-completions to the user's need as a function of their features (e.g. frequency, popularity within the same day), and the features of the user (e.g. gender, age).

While the model proposed by Li *et al.* (2014) is more general than the model we discuss in Section 4.3, these differences are less important in the considered evaluation scenario of the query auto-completions. Indeed, under the evaluation scenario we work with, the set of the user queries that are used for the evaluation is fixed. After that, we evaluate how successful a particular query auto-completion ranking system is at reducing the user's effort to submit these queries. Under such a scenario, the modelling of the relevance of the query auto-completion to the user's need is not required, as it is assumed that the user only needs to submit the target query, i.e. the query they actually submitted, with or without help of the QAC mechanism.[21] Similarly, once it is assumed that the user is only interested in submitting their target query, the top-to-bottom examination process used in (Li *et al.*, 2014) is not different from the position-based examination used in Section 4.3: the probability of examining a particular position can be represented as a sum of the probabilities the user decides to examine the list of auto-completions a depth below or equal to this particular position.

Finally, the QAC offline evaluation framework we propose is general with respect to the underlying user interaction model, and can benefit from using a more elaborated model of the user behaviour. In this chapter we use the user interaction model that is introduced in Section 4.3 as a foundation for the evaluation framework.

[21] While this assumption might be relaxed, this would require a labour-intensive labelling process. Moreover, it is unclear how useful these judgements can be in the presence of highly personalised query auto-completions used by modern search engines (Bar-Yossef & Kraus, 2011; Kharitonov *et al.*, 2013*a*; Zhang *et al.*, 2015).

4.2.3 Comparing IR Metrics

Since the aim of the evaluation is to predict whether the retrieval system meets the user needs, it is natural to require the values of evaluation metrics to be aligned with the user preferences. A considerable effort was deployed to ensure that the metrics used in web search evaluation settings meet this criterion. Moreover, this alignment is crucial to ensure the overall efficiency of the evaluation pipeline (Chapter 3), as it affects how many changes proceed to the online evaluation step and are rejected after it. However, there is no agreement in the way the user preferences should be obtained.

Some authors conducted online user-based studies to address this question. For instance, Radlinski & Craswell (2010) studied the agreement between Cranfield-based measures such as MAP and nDCG with results obtained from online user-based evaluation experiments. Sanderson *et al.* (2010) compared the outcomes of a large-scale side-by-side evaluation of retrieval systems performed by the users of the Mechanical Turk[22] with a preference relation over these systems imposed by Cranfield-based measures such as nDCG, MRR and $Precision@10$ as well as the diversity measures including α-nDCG (Clarke *et al.*, 2008), cluster recall and intent-aware precision.

ERR is supported in (Chapelle *et al.*, 2009) by a series of experiments which demonstrate that ERR shows better alignment with user behaviour in comparison with other widely used metrics. These experiments fall into two different categories. Firstly, the authors demonstrate that across different queries and possible rankings, ERR is better correlated with user click metrics such as search success rate. Secondly, in a simulated experiment it was shown that the difference in ERR of two ranking functions is better correlated with the difference in actual user preferences in comparison with other metrics. Chapelle *et al.* (2011) used a similar approach to compare various metrics used to evaluate diversified result set.

Similar ideas are used by Chuklin, Serdyukov & de Rijke (2013). In particular, they used the correlation between the offline metrics and the outcomes of online experiments (both interleaving and A/B tests) as a tool to evaluate the quality of the offline metrics.

The evaluation methodology we use in this chapter is influenced by the methods of Chapelle *et al.* (2009), in that we compare the considered metrics in terms of their correlation with the user preferences observed in historical query logs.

4.2.4 Query Auto-completion Evaluation

Shokouhi & Radinsky (2012) evaluated the quality of suggestions for a given prefix by the mean reciprocal rank of the most popular results (MRR), and the Spearman correlation between the predicted

[22] Amazon Mechanical Turk (https://www.mturk.com/) is a crowdsourcing marketplace.

and ground-truth ranks of the selected queries. These metrics were averaged over a set of test prefixes. The MRR metric is also used in (Li *et al.*, 2014; Mitra, 2015; Zhang *et al.*, 2015). Considering an auto-completion as relevant if it is top-ranked according to the ground-truth query frequencies, Strizhevskaya *et al.* (2012) reported $P@3$, $AP@3$ and $nDCG@3$ averaged over the observed prefixes.

Bar-Yossef & Kraus (2011) used a session log-based approach for evaluation, which aimed to emulate user experience. From a session in the log, they extracted the user's context and the submitted query. After that, the suggestions are filtered to have the first character of the query as a prefix and ranked according to the user's context. The quality of the ranking is assessed as a reciprocal rank of the user's query in this ranked list of suggestions, weighted by the number of completions available for the prefix. They reported the weighted mean reciprocal rank (wMRR) averaged over the query-context pairs.

Duan & Hsu (2011) used the minimal number of key presses the user has to make in order to issue the target search query (Minimal Keystrokes, MKS) to evaluate query corrections. This metric evaluates the effectiveness of the QAC mechanism with respect to the user who always selects the optimal way of submitting a query.

As we can see from the related work, the query auto-completion effectiveness metrics used in the literature are not specifically designed to reflect user satisfaction nor has their suitability been empirically shown. We address this gap in the context of query auto-completion mechanisms by firstly modelling the user behaviour in Section 4.3 and devising an effectiveness metric upon it later in Section 4.6.

In the next section, we introduce our proposed model of the user's interactions with the query auto-completion mechanism.

4.3 User Model

The interface usually used to present query auto-completion (Figure 4.1) leads to the following process of users' interaction with the query auto-completion mechanism. Let us suppose that a user is going to submit a query q to a search engine. After typing each character, the list of auto-completions is changed to match the updated input and the user has a chance to examine the list before typing the next character. The user examines the query in the jth position with some probability or skips it. If the query q is suggested by the system and examined by the user, they select it from the list and that ends the interaction with the query auto-completion mechanism. In the worst case, the user types the entire query q manually.

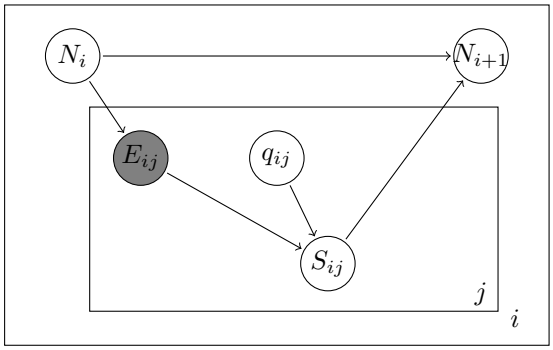


Figure 4.2: Graphical model of the user behaviour. Grey circles correspond to latent variables.

In order to build a user model upon this interaction schema, we assume that the user's behaviour satisfies the Markovian assumption, i.e. that given the user's current state, all of the future user's actions are independent from their earlier behaviour. The Markovian assumption is often used in web search behaviour models, e.g. the cascade model (Craswell *et al.*, 2008) describes the user's clicking behaviour as a Markov process.

The underlying graphical model is depicted in Figure 4.2. Denoting the prefix of the query q of length i as $q[1..i]$ and the query suggested on position j after submitting $q[1..i]$ as q_{ij}, we introduce the following binary random variables used in the graphical model:

- N_i: equals 1 if the ith character of the query q is submitted, and 0 otherwise;
- E_{ij}: equals 1 if the query suggested on position j for the prefix $q[1..i]$ (q_{ij}) is examined, and 0 otherwise;
- S_{ij}: equals 1 if the user is satisfied with q_{ij} after submitting $q[1..i]$, and 0 otherwise.

The model can be described using the following system of equations in terms of the random variables introduced above:

$$N_1 = 1 \tag{4.1a}$$

$$N_i = 0 \Rightarrow N_k = 0, k = (i+1)...|q| \tag{4.1b}$$

$$N_i = 0 \Rightarrow \forall j \; E_{ij} = 0 \tag{4.1c}$$

$$P(E_{ij}|N_i = 1) = f(i,j) \tag{4.1d}$$

$$E_{ij} = 0 \Rightarrow S_{ij} = 0 \tag{4.2e}$$

$$S_{ij} = 1 \Leftrightarrow E_{ij} = 1,\ q_{ij} = q \tag{4.2f}$$

$$\exists j : S_{ij} = 1 \Rightarrow N_{i+1} = 0 \tag{4.2g}$$

$$\forall j\ S_{ij} = 0, i < |q| \Rightarrow N_{i+1} = 1 \tag{4.2h}$$

$$|q| = i,\ \forall j\ S_{ij} = 0 \Rightarrow N_{i+1} = 0 \tag{4.2i}$$

Indeed, the above equations describe the following constraints on the model: The first character is always submitted (4.1a); Characters are submitted sequentially (4.1b); Only the suggestions for the submitted prefixes can be examined (4.1c); The probability of examining the query suggested in the jth position is a function of its position j and the length of the submitted prefix i (4.1d); A non-examined auto-completion cannot satisfy the user (4.2e); Examination of the query q is necessary and sufficient for the user to be satisfied, i.e. after examining the query q the user is always satisfied (4.2f)[23]; A satisfied user stops interaction with the query auto-completion mechanism (4.2g); An unsatisfied user types the query until its end (4.2h) & (4.2i).

In the model described above, we do not specify the exact form of dependence of examination probabilities denoted as a function $f(i, j)$. Varying the form of this dependence we can obtain different user models. For instance, the following functions can be considered:

1. The user always examines all the suggested queries: $f_1(i, j) = 1$;
2. The examination probability depends only on position j and decays under reciprocal $f_{rr}(i, j) = 1/(j + 1)$ or logarithmic $f_{log}(i, j) = 1/\log_2(j + 2)$ laws[24];
3. The examination probability depends only on the position: $f_l^i(i, j) = A_j$;
4. The examination probability depends not only on the position, but also on the prefix length: $f_l^d(i, j) = B_{ij}$;

The functions $f_l^i(i, j)$ and $f_l^d(i, j)$ depend on parameters, A_j and B_{ij}, respectively. Instead of using heuristic functions such as $f_{rr}(i, j)$ or $f_{log}(i, j)$, these parameters can be learned to obtain a model that

[23] We do not explicitly model the event of the query being seen, but not selected from the suggestions list by the user. In a such event, the query is also considered as non-examined. This does not influence the generality of our proposed model.

[24] The logarithmic and reciprocal rank decays are used in DCG (Järvelin & Kekäläinen, 2002) and MRR document search metrics, respectively. We shift both functions so that they start on the second rank position for two reasons: 1) the resulting probabilities are closer to those observed in our query log; 2) we avoid singularities in the user model evaluation in Section 4.7.1: without the shifting the model would consider it impossible for the user not to examine the first position, although it happens in the dataset.

Input: $\mathbb{A}$: a set of interactions with QAC used
Output: Examination probabilities A_j, B_{ij}
Initialise $\forall i, j\ A_j^c = 0,\ A_j^s = 0,\ B_{ij}^c = 0, B_{ij}^s = 0$
foreach $a \in \mathbb{A}$ **do**
  foreach $i \in 1..|q(a)|, j$ **do**
    if $q_{ij}(a) = q(a)$ *and* $q_{ij}(a)$ *not clicked* **then**
      $A_j^s \leftarrow A_j^s + 1$
      $B_{ij}^s \leftarrow B_{ij}^s + 1$
    end
    if $q_{ij}(a) = q(a)$ *and* $q_{ij}(a)$ *clicked* **then**
      $A_j^c \leftarrow A_j^c + 1$
      $B_{ij}^c \leftarrow B_{ij}^c + 1$
    end
  end
end
foreach j **do**
  $A_j \leftarrow \frac{A_j^c}{A_j^c + A_j^s}$
end
foreach $i,\ j$ **do**
  $B_{ij} \leftarrow \frac{B_{ij}^c}{B_{ij}^c + B_{ij}^s}$
end

Algorithm 4.1: Learning the prefix length-independent A_j and the prefix length-dependent B_{ij} probabilities of examination of a query suggestion presented on position j for a prefix of length i.

better reflects the user behaviour. In the following, we discuss an algorithm to adjust these parameters to the user behaviour observed in a session log.

The proposed model of the user behaviour is related to the cascade model (Craswell *et al.*, 2008) of the user's search click behaviour. Indeed, the user continues to type their query if and only if they are not satisfied with the current suggested queries. On the other hand, the process of examination of the list of query auto-completions resembles that considered in the position-based user models (Craswell *et al.*, 2008).

4.4 Learning the Model Parameters

Let us consider a QAC interaction with a query q submitted by selecting it from the list of queries suggested to a prefix $q[1..l]$ on position k. Before the interaction stopped, the following random events took place. The user skipped the query q each time it was suggested for a prefix shorter than l. The probability of this is equal to the following expression:

$$P_{skip} = \prod_{i=1}^{l} \left[1 - \sum_{j} \mathbb{I}(q = q_{ij}) P(E_{ij}) \right] \tag{4.3}$$

where $\mathbb{I}(q = q_{ij})$ equals 1 if the query q was suggested on position j for the prefix $q[1..i]$, and 0 otherwise. Further, the user examined the kth suggested query for prefix $q[1..l]$. Thus, the likelihood of the entire interaction a is:

$$L(a) = \prod_{i=1}^{l} \left[1 - \sum_{j} \mathbb{I}(q = q_{ij}) P(E_{ij}) \right] P(E_{lk}) \tag{4.4}$$

By $l(a)$ we denote the length of the prefix typed in an interaction a. Substituting $P(E_{ij})$ with $f(i, j)$ the log-likelihood of a set of interactions $\mathbb{A}$ can be represented in the following form:

$$\hat{L} = \sum_{a \in \mathbb{A}} \sum_{i=1}^{l(a)} \sum_{j} \log \left[(1 - \mathbb{I}(q_{ij} = q) f(i, j))^{1 - S_{ij}} \cdot f(i, j)^{S_{ij}} \right] \tag{4.5}$$

The log-likelihood expressed in Equation (4.5) can be maximised with respect to the function f to find maximum likelihood estimates (MLE) of its parameters. $P(E_{ij})$ are Bernoulli distributed random variables with their probabilities of success (i.e. examination probability) determined by the prefix length-independent f_l^i or the prefix length-dependent f_l^d functions, discussed in the previous section. We can find the estimates of their parameters, A_j and B_{ij}, using the maximum likelihood principle, as described in Algorithm 4.1, which resembles the learning process for obtaining the parameters of the cascade model (Craswell *et al.*, 2008).

The functions f_l^i and f_l^d with parameters A_j and B_{ij}, respectively, optimised on a training part of a dataset described in Section 4.8 are reported in Table 4.1. We additionally report values of f_{rr} and f_{log} for comparison. When calculating the prefix length-dependent examination probability function f_l^d we assume that the examination probabilities for prefixes longer than six characters become independent from the prefix length.

Based on an analysis of Table 4.1, the following conclusions can be made. Firstly, the probability of examination E_{ij} shows considerable dependence on the prefix length i: for shorter prefixes, the users tend to examine the suggested queries more carefully. For instance, the probability of examination of the first position changes from $f_l^d(1, 1) = 0.50$ for prefixes of length 1 to $f_l^d(6, 1) = 0.20$ in case of prefixes of length 6. Another observation is that the probabilities of examination for a fixed position become almost equal for prefixes of length five and six, e.g. $f_l^d(5, 1) = 0.19 \approx f_l^d(6, 1) = 0.20$ and $f_l^d(5, 10) = 0.14 \approx f_l^d(6, 10) = 0.15$. This observation justifies our decision to model the examination probabilities as being independent from the prefix length for longer inputs.

Even if one considers the approximation of the probabilities of examination to be independent from the prefix length, the resulting probabilities estimated from the query logs (the function f_l^i) are different

Table 4.1: Probability of examination. $f_{rr}(i,j)$, $f_{log}(i,j)$ and $f_l^i(i,j)$ correspond to the prefix length-independent logarithmic, reciprocal, and learned examination probabilities, respectively. $f_l^d(i,j)$ denotes the prefix length-dependent probabilities of length i, as learned from the query log.

	Query rank, j									
	1	2	3	4	5	6	7	8	9	10
$f_{rr}(\cdot,j)$	0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09
$f_{log}(\cdot,j)$	0.63	0.50	0.43	0.39	0.36	0.33	0.32	0.30	0.29	0.28
$f_l^i(\cdot,j)$	0.21	0.19	0.17	0.16	0.15	0.14	0.14	0.14	0.14	0.13
$f_l^d(1,j)$	0.50	0.28	0.23	0.26	0.20	0.17	0.18	0.17	0.15	0.15
$f_l^d(2,j)$	0.46	0.25	0.26	0.23	0.19	0.17	0.16	0.16	0.15	0.12
$f_l^d(3,j)$	0.21	0.19	0.17	0.16	0.15	0.14	0.14	0.13	0.13	0.12
$f_l^d(4,j)$	0.23	0.22	0.19	0.18	0.16	0.16	0.15	0.15	0.15	0.14
$f_l^d(5,j)$	0.19	0.20	0.17	0.16	0.15	0.15	0.15	0.15	0.15	0.14
$f_l^d(6,j)$	0.20	0.21	0.19	0.19	0.17	0.16	0.16	0.15	0.16	0.15
$f_l^d(7,j)...f_l^d(\infty,j)$	0.14	0.16	0.15	0.14	0.13	0.13	0.13	0.13	0.13	0.12

from the one imposed by the position discount functions often considered in other domains of Information Retrieval: the first two positions have considerably lower chances to be examined ($f_l^i(1,1) = 0.21$) than it is predicted by the f_{log} ($f_{log}(1,1) = 0.63$) or f_{rr} ($f_{rr}(1,1) = 0.5$) functions. Also, it is noticeable that the reciprocal rank function decays with the suggestion rank faster than the examination probabilities learned from the user behaviour in the session log ($f_{rr}(1,10) = 0.09$ while $f_l^i(1,10) = 0.13$).

We have introduced the user model, its possible instantiations and the algorithm to learn their parameters from the session log. Before defining the proposed user model-based effectiveness metrics in Section 4.6, we firstly describe the evaluation framework we are working within and which we define in Section 4.5.

4.5 Offline Evaluation of Query Auto-completion Mechanism

Before defining new effectiveness metrics for the query auto-completion evaluation we need to discuss our evaluation framework used to evaluate a query auto-completion mechanism. We use the same query log-based approach as used in (Bar-Yossef & Kraus, 2011). In this section, we discuss the evaluation scenario imposed by the framework, its restrictions and how it compares to the experimental methodologies used in the query auto-completion literature, discussed in Section 4.2.4.

In the case of the query auto-completion domain, the offline evaluation approach implies the following evaluation algorithm. Given a query auto-completion mechanism with a ranking function r and a log of queries $\mathbb{Q}$, the evaluation is performed in three steps. At the first step, the process of submitting

a query $q \in \mathbb{Q}$ is simulated as if a user typed it. For each prefix $q[1..i]$, all of the possible candidate suggestions are ranked according to the ranking function r, i.e. the simulated user is presented with a ranked list of suggestions $r(q[1..i])$. Considering q as the target query, i.e. the only query the user wants to submit, this simulated output is used to estimate the effectiveness metric. Finally, the metric is averaged over all simulated interactions.

In order to perform such an evaluation, a query log is needed. However, we consider this requirement as not restrictive in the query auto-completion effectiveness assessment, since query auto-completion mechanisms are often built upon the query log mining (Bar-Yossef & Kraus, 2011).

We believe that this approach is sufficiently general to cover several interesting evaluation scenarios, e.g. the minimal dataset required to evaluate the query auto-completion mechanism includes only queries (and possibly their frequencies). In the more advanced setting, the same methodology is suitable to evaluate personalised ranking algorithms by associating each session with its context (Bar-Yossef & Kraus, 2011; Kharitonov *et al.*, 2013*a*; Shokouhi & Radinsky, 2012; Strizhevskaya *et al.*, 2012) or the user's long-term profile. It is also noticeable that this evaluation scenario generalises other approaches to evaluate query auto-completion mechanisms discussed in the literature (Section 4.2.4). For instance, in (Shokouhi & Radinsky, 2012; Strizhevskaya *et al.*, 2012) the test sets of prefixes are sampled and the considered systems are evaluated by assessing how good the most popular prefix completion was ranked having this prefix as the user input. It is possible to consider their methodology as a special case of the evaluation approach used in this chapter. Indeed, considering the set of popular queries for all the test prefixes, one can generate simulated sessions with users submitting these prefixes and measuring the system effectiveness afterwards.

Finally, we want to highlight that this evaluation scenario is akin to the one used in the Cranfield paradigm (Section 2.2), as it abstracts the ranker from other systems and model the user's interaction with the system.

Guided by this evaluation scenario, in the next section we introduce novel evaluation metrics for query auto-completions.

4.6 Proposed Metrics

The effort-based metrics (Chuklin, Serdyukov & de Rijke, 2013) and the cascade-based metrics (Chapelle *et al.*, 2009) are defined as the expectation of an effort[25] function at the position the user finds the result

[25] In (Chapelle *et al.*, 2009) such a function is referred to as an utility function, however, to avoid confusion with the utility-based metrics considered in a more recent work (Chuklin, Serdyukov & de Rijke, 2013) we adopt the terminology from (Chuklin, Serdyukov & de Rijke, 2013).

they are looking for. In order to generalise this definition to the query auto-completion evaluation, let us recall the notation previously used in Section 4.3.

We denote the query to be submitted as q and its length as $|q|$. A prefix of q of length i is referred to as $q[1..i]$. E_{ij} is a binary variable equal to 1 if a query suggestion for the prefix $q[1..i]$ ranked in the jth position is examined by the user, S_{ij} is a binary variable representing if the user was satisfied with the jth suggestion shown for the prefix $q[1..i]$. q_{ij} denotes a suggested query ranked on position j after submitting $q[1..i]$. We denote the effort function as $U(i, j)$.

Using this notation, we can adapt the notion of the effort-based metric $V(q)$ to the query auto-completion evaluation:

$$V(q) = \sum_{i=1}^{|q|} \sum_{j} U(i,j) P(S_{ij} = 1) \tag{4.6}$$

where $\sum_j P(S_{ij} = 1)$ equals to the probability to stop immediately after submitting $q[1..i]$. $P(S_{ij})$ represents the probability to stop immediately after submitting $q[1..i]$ (Section 4.3), and it depends on the positions where the query q is suggested and the examination probabilities (Table 4.1).

The question arises how to choose the effort function $U(i, j)$. In the simplest case, one can define the effort function to be the unity constant. Given such an effort function, the metric equates to the probability of the user using the query auto-completion mechanism. We refer to this metric as *pSaved*, and it is formally defined as follows:

$$pSaved(q) = \sum_{i=1}^{|q|} \sum_{j} P(S_{ij} = 1) \tag{4.7}$$

A similar utility function was used in (Chapelle *et al.*, 2009) to build the modification of ERR based on the cascade model with abandonment. Since the *pSaved* metric represents the probability of satisfying the user, its values are non-negative and below or equal to 1.

A more complicated function $U(i, j)$ might decrease if it takes the user more effort[26] to find a satisfactory result and thus it can be considered as a formalisation of the amount of the effort the user can avoid due to the retrieval system under consideration. In the case of a query auto-completion mechanism, the user's effort can naturally be represented as the number of characters (keypresses) the user has to type to submit their query to the system. Supported by this intuition, we propose the metric *eSaved*, which is defined as the expected ratio of characters a user can skip inputting until their query is submitted. The query can be submitted either by selecting it from the suggested list of queries or by fully entering the

[26]The effort function used in ERR degrades as the user examines more retrieved results.

query. Formally, *eSaved* can be calculated using the following expression:

$$eSaved(q) = \sum_{i=1}^{|q|} \left(1 - \frac{i}{|q|}\right) \sum_{j} P(S_{ij} = 1) \tag{4.8}$$

where $U(i,j) = \left(1 - \frac{i}{|q|}\right)$ is the effort function.

Both proposed metrics, *pSaved* and *eSaved* are parameterised by the user model, which defines the probability of the user satisfaction $P(S_{ij} = 1)$. In the user model proposed in Section 4.3, this probability is defined by Equations (4.1d), (4.2e) and (4.2f). The user is satisfied with a suggested query q_{ij}, only if it is the target query ($q_{ij} = q$) and if it is also examined ($E_{ij} = 1$):

$$S_{ij} = 1 \Leftrightarrow E_{ij} = 1, q_{ij} = q$$

In turn, the examination probability is determined by the user attention function $f(i,j)$. Using the definition of the proposed metrics, we can calculate them by Algorithm 4.2. Efficient, linear-time approaches to calculate the *Saved* metrics on sets of queries were proposed in (Loptev *et al.*, 2014).

In order to get an additional insight into the difference between the proposed metrics, *pSaved* and *eSaved*, we re-group Equation (4.8) in the following form:

$$eSaved(q) = \sum_{i=1}^{|q|} \sum_{j} P(S_{ij} = 1) - \sum_{i=1}^{|q|} \frac{i}{|q|} \sum_{j} P(S_{ij} = 1) \tag{4.9}$$

Comparing (4.7) and (4.9) we notice that *eSaved* equals to *pSaved* minus the expected part of the query the user needs to type to submit query q. As a result, *eSaved* additionally stratifies queries with equal chances to satisfy the user, according to the relative length of the query the user needs to type.

This ability of $eSaved$ to leverage this additional "dimension" to assess the QAC ranking functions can be particularly useful for longer queries where its utility function has a wide spectrum of values. We believe that improvements in the query auto-completions for longer queries have a high influence on the user experience. Indeed, when submitting a long query, the auto-completion mechanism can save greater effort for the user than in the case of a short query.

In this section, we proposed two novel metrics to evaluate the effectiveness of the query auto-completion mechanisms. However, it is unclear how one can compare QAC effectiveness metrics and in the next section we discuss this issue.

4.7 Experimental Methodology

Our empirical study has the following goals. The first goal is to investigate how the user model instantiations introduced in Section 4.3 compare to each other in terms of fitness to the observed user behaviour

Input: Query q; lists of query auto-completions q_{ij} for each prefix $q[1..i]$; effort function $U(i,j)$; user attention function $f(i,j)$.
Output: The value of the metric for a query q, $V(q)$.
// The value of the metric
$V(q) \leftarrow 0$
// The probability that the user was not satisfied so far
$p_{cont} \leftarrow 1$
foreach $i \in 1..|q|$ **do**
  //is q suggested among the auto-completions?
  if $\exists j: \ q_{ij} = q$ **then**
    $V(q) \leftarrow V(q) + p_{cont} \cdot f(i,j) \cdot U(i,j)$
    $p_{cont} \leftarrow p_{cont} \cdot (1 - f(i,j))$
  end
end

Algorithm 4.2: Calculating the value of a metric from the *Saved* family.

in the data. Each of these user models induces a corresponding metric of the *Saved* family and the question arises how well these metrics are aligned with the user behaviour data. It is also important to compare the proposed metrics with the ones previously used in the literature. We formulate these goals as three research questions (RQ):

RQ4.1 Which of the considered examination probability functions f_{rr}, f_{log}, f_l^i, and f_l^d better "explain" the user behaviour observed in the data?

RQ4.2 How do our proposed $Saved$ metrics compare to the metrics used in the literature?

RQ4.3 Which metrics can be recommended for the evaluation of the QAC mechanisms?

In order to answer these questions we use the methodology described below. In Section 4.7.1, we discuss how we compare the considered user models. In Section 4.9.2, we discuss how to evaluate the considered metrics. We discuss the baseline metrics used in the literature in Section 4.7.3.

4.7.1 User Model Evaluation

The effectiveness of the user models is often studied by means of measuring the log-likelihood on the test data, e.g. (Chuklin *et al.*, 2015; Zhang *et al.*, 2011). The log-likelihood is defined as the average log probability of the events observed in the test dataset according to the probabilistic model under consideration.

Let a be an interaction from a dataset of QAC interactions $\mathbb{A}$, $q(a)$ denotes the query submitted by the user in the interaction a and C_a^i is a binary indicator representing if the user's interaction with the QAC mechanism ended (i.e. no additional characters were typed) after submitting the prefix $q(a)[1..i]$. By

definition, $C_a^{|q(a)|} = 1$ if the user typed the entire query. Again, $l(a)$ denotes the number of characters submitted during the interaction a.

In the case of the query auto-completion, the log-likelihood of the user model measures how well this model predicts a prefix which the user typed before submitting the query. More formally, the log-likelihood of the model on the session with the submitted query q is defined in the following way:

$$L(a) = \sum_{i=1}^{l(a)} \left[P(C_a^i) \log_2 P(C_a^i) + (1 - C_a^i) \log_2 (1 - P(C_a^i))\right]$$

The overall log-likelihood is calculated as the average of the session likelihoods:

$$LL(\mathbb{A}) = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} L(a)$$

Another measure widely used to evaluate the performance of click models is the average *perplexity* for top ten positions (Dupret & Piwowarski, 2008). However, since the queries differ in their length and the number of auto-completions available, the perplexity becomes less intuitive in the considered case.

4.7.2 Metrics Evaluation

As we discussed in our roadmap for improving the evaluation pipeline in Section 3.5, our goal in improving the offline evaluation step is to increase its alignment with the outcomes of the online evaluation step. This alignment ensures that more online experiments would be successful, thus increasing the efficiency of the evaluation pipeline as a whole. Consequently, it is reasonable to compare the considered metrics in terms of their correlation with the online preferences of users.

A similar evaluation methodology was used in (Chapelle *et al.*, 2009; Chuklin, Serdyukov & de Rijke, 2013; Markov *et al.*, 2014). This methodology is aimed to show how good the tested metrics are aligned with the online user satisfaction indicators. Chapelle *et al.* (2009) considered different click measures as indicators of the user interest, such as the search abandonment rate, the position of the first click and others. We believe that the query auto-completion mechanism can be considered as useful and successful in a particular session if the user used it to submit their query. Thus we use the QAC usage frequency (how often the query auto-completion mechanism is used by users to submit their queries) as a ground-truth indicator of the user satisfaction. In the following, we refer to this value as *success rate* (S). In general, other indicators of user satisfaction can be considered. The indicator selected might influence the evaluation result and should be chosen in agreement with the QAC performance indicators. We use *S* as it is readily available in the query logs, widely used as a metric in document search A/B experiments[27] and is easy to interpret in the context of QAC.

[27] In document search, this metric is typically called *click-through rate*, it is the opposite of the abandonment rate metric, discussed in Section 2.2.

Due to various personalisation algorithms, geographical contextualisation features, drifts in query popularity and changes in auto-completion mechanism, the positions where the target query is presented can vary. By configuration c we denote a unique tuple of the query q and the sequence of positions where q is suggested in the list of query auto-completions while q is typed. This sequence has length of $|q|$. If for a particular prefix length the query is ranked below 10th position it is considered as not demonstrated. When building a configuration such an event is encoded by a sentinel value.

The considered metric evaluation method is performed in two steps. Firstly, given a dataset of user interactions one can calculate the values of the considered metrics for each configuration observed by a user. On the other hand, for each configuration shown to the users the average value of the success rate can be estimated. In the next step, the correlation between these two values across configurations is calculated. In this chapter, we use weighted correlation proposed by Chapelle *et al.* (2009) and also used in (Chuklin, Schuth, Hofmann, Serdyukov & de Rijke, 2013; Chuklin, Serdyukov & de Rijke, 2013; Markov *et al.*, 2014). This weighted correlation is akin to the standard Pearson correlation of online and offline metrics across configurations. However, each configuration is additionally weighted according to its frequency in the query stream. This weighting allows the correlation to reflect the relative importance of the queries (Chapelle *et al.*, 2009) and reduces the effects of the noise in estimates of the success rate for the rare queries.

By N we denote the total number of configurations, n_i denotes the number of interactions with ith configuration in the dataset. M_i is the value of the offline evaluation metric M calculated on for the ith configuration, and the mean of the online satisfaction indicator S is denoted by S_i. Then the weighted correlation between M and S is defined as follows:

$$C(M,S) = \frac{\sum_{i=1}^{N} n_i (M_i - \bar{m})(S_i - \bar{s})}{\sqrt{\sum_{i=1}^{N} n_i (M_i - \bar{m})^2} \sqrt{\sum_{i=1}^{N} n_i (S_i - \bar{s})^2}} \tag{4.10}$$

where $\bar{m}$ and $\bar{s}$ are weighted means of the tested offline metric and the online indicator:

$$\bar{m} = \frac{1}{\sum_{i=1}^{N} n_i} \sum_{i=1}^{N} n_i M_i, \quad \bar{s} = \frac{1}{\sum_{i=1}^{N} n_i} \sum_{i=1}^{N} n_i S_i$$

As discussed by Chapelle *et al.* (2009), this approach has one possible drawback. Indeed, the correlation measures the alignment of the metrics with the user satisfaction across configurations and queries, i.e. it also measures how useful the metric is to compare the effectiveness of different queries. However, in a real-life scenario, the primary goal of a metric is to compare different ranking functions of query auto-completions given a fixed set of queries. Therefore, another metric's feature is essential: if, given a *fixed* set of queries one ranking algorithm outperforms its counterpart in terms of the considered metric,

does this necessarily imply that the first algorithm has higher user satisfaction once deployed? There is a possibility that a particular metric can exhibit a poor performance when comparing effectiveness across queries but still exhibits a good alignment with the user satisfaction from the ranking function comparison perspective.

In order to study the metric quality from the latter point of view, we perform an experiment similar to one proposed in (Chapelle *et al.*, 2009), which simulates a scenario of an A/B test-based comparison of ranking functions r_1 and r_2. This simulation is performed according to Algorithm 4.3. The idea behind the algorithm is as follows. For each query with at least two configurations in the session log, configurations c_1 and c_2 are randomly sampled. These configurations and the corresponding sessions are associated with two simulated ranking functions, r_1 and r_2, as if all sessions with c_1 and c_2 shown to the users were served by ranking algorithms r_1 and r_2, respectively. After that, the average values of the considered metric M and the user satisfaction indicator S are calculated for both systems. We additionally weight the values of the metrics to reflect the actual distribution of the queries in A/B test. This weighting ensures that (a) the relative importance of the queries is preserved in our simulation, and (b) each query has equal impact on both simulated alternatives r_1 and r_2. After iterating over all queries in the dataset, we calculate the differences of the user satisfaction indicators $S_1 - S_2$ and the effectiveness metric values $M_1 - M_2$ for r_1 and r_2. By repeating this simulation, we generate a set comparisons of pairs of systems. Finally, we calculate the correlation between the differences in the offline metric and the differences in satisfaction indicator. Higher correlation implies better agreement between offline and online metrics when comparing different rankings when the query stream is fixed and it is the alignment we want to increase to improve the efficiency of the evaluation pipeline (Section 3.5).

Overall, in this section we discussed two possible ways to quantify the agreement between an offline effectiveness metric and an online satisfaction indicator, namely weighted correlation (Equation 4.10) and A/B test simulation (Algorithm 4.3).

After discussing how to compare offline metrics, in the next section we discuss the baseline QAC effectiveness metrics we use in our study in this chapter.

4.7.3 Baseline Metrics

In our evaluation study, we use the following baseline offline QAC evaluation metrics used in the literature:

MRR-n is a metric which is defined as the reciprocal rank of the submitted query q after submitting the first n characters of the query. For queries shorter than n characters we define *MRR-n* to be equal to *MRR-*$|q|$. Ranks higher than 10 are considered to be infinitely large (i.e., *MRR-1* and *MRR-3* are equal

Input: $\mathbb{Q}$: a dataset of queries
Output: Correlation between the QAC success rate S and an effectiveness metric M
foreach $i \in 1..1000$ **do**
  Simulating two search engines, r_1 and r_2
  Initialise online and offline metrics counters
  $S_1 \leftarrow 0, S_2 \leftarrow 0$
  $M_1 \leftarrow 0, M_2 \leftarrow 0$
  foreach *query* $q \in \mathbb{Q}$ **do**
    $C_q \leftarrow$ set of configurations of q
    Initialise the total number of interactions in simulated systems
    $n \leftarrow 0$
    if $|C_q| \geq 2$ **then**
      $c_1, c_2 \leftarrow$ two random configurations from C_q
      Assign c_1 to a simulated system r_1, c_2 to a simulated system r_2.
      Weight the values of the offline metric M according to the query frequency $freq(q)$.
      $M_1 \leftarrow M_1 + M(c_1) \cdot freq(q)$
      $M_2 \leftarrow M_2 + M(c_2) \cdot freq(q)$
      Weight the mean of success rates $S(c1)$ and $S(c_2)$ for configurations c_1 and c_2.
      $S_1 \leftarrow S_1 + S(c_1) \cdot freq(q)$
      $S_2 \leftarrow S_2 + S(c_2) \cdot freq(q)$
      Update the number of interactions
      $n \leftarrow n + freq(q)$
    end
  end
  Compute the mean values of M and S:
  $M_1 \leftarrow \frac{1}{n} M_1$, $M_2 \leftarrow \frac{1}{n} M_2$
  $S_1 \leftarrow \frac{1}{n} S_1$, $S_2 \leftarrow \frac{1}{n} S_2$
end
Return Pearson correlation between differences in $M_1 - M_2$ and $S_1 - S_2$ for all simulated pairs of r_1 and r_2

Algorithm 4.3: Algorithm used to measure the correlation of the difference in an offline effectiveness metric M with the differences in online user satisfaction indicator S in simulated A/B tests.

to 0 if the submitted query is ranked on positions below 10). The MRR metric is used in (Shokouhi & Radinsky, 2012), as discussed in Section 4.5.

By *wMRR-n* we denote a modification of *MRR-n* weighted by the number of suggestions available for the corresponding query prefix, as used by Bar-Yossef & Kraus (2011). Such a weighting is aimed to promote systems that perform better in "hard" cases, where a lot of candidate queries are available.

negMKS Minimal Keystrokes (MKS) is a metric proposed by Duan & Hsu (2011) to evaluate the query misspelling correction algorithms, which is defined as the minimal number of keystrokes a user has to perform in order to submit the query. The minimum is calculated among all possible interactions: the user can type the query's next character or can select the query from the list of suggested auto-completions using arrow keys. Despite the fact that it was proposed to evaluate misspelling correction

Table 4.2: QAC dataset statistics.

Sessions	Unique configuration	Unique queries	Mean length, characters	Median length	Mean terms	Median terms
6.1M	3.8M	3.3M	26.4	24.0	3.3	4.0

algorithms, MKS can also be used to evaluate query auto-completions. By definition, a better system should have lower MKS, thus the correlation between the user satisfaction indicator and MKS should be negative. Hence, to ensure the uniformity of the results and make their comparison more intuitive, we use the negative of MKS as a metric. We denote it as $negMKS$.

4.8 Dataset

Before discussing the experimental results in the next section, we shortly describe the dataset used in chapter. The dataset was randomly sampled from a log of queries submitted to Yandex. The search sessions were performed by users from Ukraine, a country with two languages, Russian and Ukrainian, widely spoken.[28] The dataset spans two consecutive workdays in January 2013. In order to reduce noise in our evaluation, we applied the following filtering procedure to the dataset. Firstly, we do not consider sessions with misspelled queries[29], leaving the adaptation of the proposed models to misspelled queries as a direction of future work. We also removed sessions where users selected a query from the suggested list and edited it afterwards since it is unclear if the query auto-completion mechanism was useful in these sessions. All queries were normalised to the lower case, since the character capitalisation is typically ignored by query auto-completion mechanisms. Finally, only query sessions with the query auto-completions shown were sampled. We report the descriptive dataset statistics in Table 4.2. The length of the query auto-completions lists was restricted by the deployed query auto-completions mechanism to be no longer than 10.

The sessions from the first day were used to estimate the user model parameters discussed in Section 4.3, while the evaluation of the models and the effectiveness metrics comparison were performed on the subset of sessions performed on the second day.

4.9 Results and Discussion

We split our evaluation experiments into two parts. Firstly, we discuss the experimental evaluation of the user models (Section 4.9.1). After that, we report the results of the effectiveness metrics evaluation

[28] We believe that results we report in this chapter generalise across different languages. However, we leave the experimental verification of this assumption as future work.

[29] We consider a query as misspelled if a proprietary spelling correction algorithm replaced the submitted query with an auto-corrected one. In our experimental study, such queries formed about 10% of the initial dataset.

Table 4.3: Log-likelihood of the user models parameterised by different examination probability functions. A higher log-likelihood (i.e. a lower absolute value of log-likelihood) corresponds to a better fit to the data. In each row, the function that demonstrates the statistically significantly highest log-likelihood (paired t-test, $p < 0.001$) is labelled with $\triangle$.

Query length	f_{rr}	f_{log}	f_l^i	f_l^d
> 0	-2.15	-2.73	-1.82	-1.76$\triangle$
1 - 10	-1.66	-2.05	-1.45	-1.32$\triangle$
11-20	-2.26	-2.78	-1.93	-1.85$\triangle$
21-30	-2.33	-2.95	-1.95	-1.94$\triangle$
> 30	-2.21	-2.97	-1.82	-1.81$\triangle$

(Section 4.9.2).

4.9.1 User Model Evaluation

The results of the user model evaluation on the test dataset are reported in Table 4.3. We report the log-likelihood of the models with the following functions determining the probability of examination of the query auto-completions discussed in Section 4.3: f_{rr}, f_{log}, f_l^i and f_l^d. The first three functions correspond to the probability examination functions which are independent from the prefix length, while the last one is the prefix length-dependent. The parameters of f_l^i and f_d^l are learned from the train dataset by means of Algorithm 4.1. We report the log-likelihood scores for different groups of queries according to their length. The values labelled by $\triangle$ correspond to functions that perform statistically significantly better than others in the same row.

As seen from Table 4.3, the models with the probabilities of examination learned from the query log (f_l^i and f_l^d) exhibit a better fit than the models parameterised by the heuristic functions on every subset of queries considered. In particular, these results answer RQ4.1: f_l^d shows the best fitness to the whole dataset.

Overall, we conclude that adjusting the model parameters to the user behaviour in the session log leads to statistically significant improvements in the model's ability to "explain" the data in the dataset. The question now is whether this improvement leads the user model-based effectiveness metrics to have a higher alignment with the user preferences. We study this question in the next section.

4.9.2 Metrics Evaluation

To report the results of the metric evaluation experiments, we use the following notation. $pSaved(f)$ corresponds to a metric of the $pSaved$ family, parameterised by function f, e.g. $pSaved(f_l^i)$ is a metric obtained by assuming the model of user behaviour with probabilities of examination determined by

Table 4.4: Weighted correlation of the effectiveness metrics with the QAC success rate. In each column $\triangle$ denotes the values that statistically significantly outperform other in the same column ($p < 0.01$, bootstrap test). In bold are the highest values in the corresponding columns.

	Query length				
	> 0	1...10	11...20	21...30	> 30
MRR-1	0.371	0.636	0.438	0.305	0.270
MRR-3	0.456	0.692	0.549	0.431	0.389
wMRR-1	0.290	0.428	0.343	0.234	0.216
wMRR-3	0.352	0.420	0.415	0.334	0.310
negMKS	0.524	0.482	0.728	0.836	0.778
$eSaved(f_{rr})$	0.863	0.828	0.818	0.875	0.921
$eSaved(f_{log})$	0.873	**0.832** $\triangle$	0.835	0.888	0.929
$eSaved(f_l^i)$	0.852	0.824	0.815	0.868	0.917
$eSaved(f_l^d)$	0.836	0.804	0.781	0.849	0.908
$pSaved(f_{rr})$	0.881	0.779	0.838	0.922	0.969
$pSaved(f_{log})$	0.865	0.731	0.827	0.919	0.969
$pSaved(f_l^i)$	**0.904**$\triangle$	0.822	0.861	0.930	0.970
$pSaved(f_l^d)$	0.904	0.814	**0.865**$\triangle$	**0.931**$\triangle$	**0.971**$\triangle$

function f_l^i. Similarly, $eSaved(f_{rr})$ is the $eSaved$ metric parameterised by f_{rr}. *MRR-1* and *MRR-3* (*wMRR-1* and *wMRR-3*) are instantiations of the *MRR* (*wMRR*) baseline metric. *negMKS* denotes the negative of the Minimal Keystrokes metric. These metrics are discussed above in Section 4.7.3.

In Table 4.4, we report the weighted correlation (Equation 4.10) of the effectiveness metrics with the query auto-completion success rate across different configurations. We also report the correlation for queries of different length. In order to do this, the queries are split into four bins according to their length: less than 10 characters long; from 10 to 20 characters; from 20 to 30, and a set of queries longer than 30 characters. In addition, we report the correlation on the entire test dataset (length > 0).

On analysing Table 4.4, we observe that all eight combinations of the proposed metrics ($pSaved$ and $eSaved$) and considered examination probability functions (f_{rr}, f_{log}, f_l^i, and f_l^d) perform better than the baseline metrics (*MRR*, *negMKS*, and *wMRR*) on each considered subset of queries ($p \leq 0.01$). Comparing $pSaved$ and $eSaved$ we see that the $pSaved$ metric is better correlated with the success rate. Moreover, this observation holds for both machine-learned functions f: $pSaved(f_l^i)$ outperforms $eSaved(f_l^i)$ and $pSaved(f_l^d)$ outperforms $eSaved(f_l^d)$ ($p \leq 0.01$). Considering the $pSaved$ metric, we notice that both machine-learned functions, f_l^i and f_l^d lead to metrics that is much better aligned with the user preferences than metrics induced by heuristic functions f_{rr} and f_{log} in each query bin ($p < 0.01$).

Table 4.5: Correlation of the difference in effectiveness metrics with the difference in the QAC success rate in the simulated A/B tests (Algorithm 4.3). In bold are the highest values in the corresponding columns.

	Query length				
	> 0	1...10	11...20	21...30	> 30
MRR-1	0.460	0.465	0.449	0.439	0.436
MRR-3	0.559	0.562	0.420	0.541	0.539
wMRR-1	0.413	0.415	0.407	0.393	0.391
wMRR-3	0.471	0.478	0.348	0.458	0.456
negMKS	0.784	0.769	0.746	0.791	0.776
$eSaved(f_{rr})$	0.805	0.793	0.732	0.798	0.791
$eSaved(f_{log})$	0.810	0.798	0.739	0.803	0.795
$eSaved(f_l^i)$	0.806	0.793	0.731	0.800	0.791
$eSaved(f_l^d)$	0.803	0.790	0.745	0.797	0.789
$pSaved(f_{rr})$	0.792	0.777	0.699	0.783	0.773
$pSaved(f_{log})$	0.794	0.778	0.700	0.784	0.773
$pSaved(f_l^i)$	0.807	0.793	0.723	0.798	0.789
$pSaved(f_l^d)$	**0.820**	**0.806**	**0.754**	**0.813**	**0.804**

The experiments reported in Table 4.4 highlights that among the studied metrics, $pSaved$ is the most suitable metric to compare the effectiveness of the query auto-completion mechanism across queries and configurations. The $pSaved$ metric parameterised with the prefix length-independent examination probability function f_l^i achieves the highest weighted correlation among the studied metrics when the entire dataset is considered. However, f_l^d performs only marginally worse (the difference is approximately $5 \cdot 10^{-4}$). In three out of four bins of queries, $pSaved$ metrics achieve higher correlation than $eSaved$ metrics. Only for shorter queries with length between 1 and 10 characters, $eSaved(f_{log})$ achieves the highest correlation.

Next, the *negMKS* demonstrated the highest alignment among the considered baselines. This is intuitive, as *negMKS* takes into account the quality of the query ranking across all possible prefixes, unlike other baselines. We observe that the weighted variants *wMRR* of the *MRR* metrics perform worse than un-weighted variants. This can be explained by the fact that the number of available query auto-completions for a particular prefix have little connection with the user experience. Consequently, this additional weighting adds noise and reduces the correlation level.

In Table 4.5, we report the correlation of the difference in effectiveness metrics with the difference in QAC success rate, measured simulated A/B experiments performed by Algorithm 4.3. Similarly to Table 4.4, we split the queries into bins according to their length.

From Table 4.5, we again observe that the proposed effectiveness metrics outperform the baseline metrics when the entire dataset is considered ($pSaved(f_l^d)$ outperforms ***MRR***, ***negMKS***, and ***wMRR***, $p < 0.05$). However, in contrast to the previous experiment, there is no statistically significant difference between the $eSaved$ and $pSaved$ metrics. The $pSaved$ metric parameterised by f_l^d demonstrates the highest overall correlation. Bearing in mind that in terms of the weighted correlation (Table 4.4) it was the second best metric after $pSaved(f_l^i)$ and their difference in these two experiments is only marginal, we conclude that the $pSaved$ metrics instantiated with machine-learned examination probability functions f_l^i and f_l^d can be considered as the recommended metrics for the QAC evaluation.

Similarly to Table 4.4, ***negMKS*** proved to be the strongest baseline in Table 4.5. Indeed, for the queries longer than 20 characters its performance is close to performance of the $Saved$ metrics and is not statistically significantly different.

Overall, the above comparisons of our proposed and the baseline metrics allows us to answer the last two research questions, RQ4.2 and RQ4.3. Indeed, in both experiments we observed that our proposed metrics achieved the highest agreement with the online satisfaction indicator, success rate (RQ4.2). Next, the $pSaved$ metrics instantiated with machine-learned examination probabilities f_l^i and f_l^d demonstrated the highest scores in both evaluation scenarios we considered (weighted correlation, Table 4.4, and A/B tests simulation, Table 4.5). This answers RQ4.3.

Overall, our experimental results suggest that the proposed metrics are better aligned with the user behaviour observed in the session log than other metrics considered, supporting the user model-based approach to build an effectiveness metric as being promising. Finally, the $pSaved$ metric is shown to be the most suitable metric to compare QAC ranking performances both across queries and when the set of queries is fixed.

4.10 Conclusions

In this chapter, we discussed a model of the user interactions with the query auto-completion mechanisms. We described a machine learning approach to adapt the model parameters to the user behaviour observed in a session log. Using the described model, we introduced two user model-based evaluation metrics, $pSaved$ and $eSaved$. The first metric, $pSaved$ is defined as a probability of using a query auto-completion mechanism while submitting a query. $eSaved$ equates to the normalised amount of keypresses a user can avoid due to the deployed query auto-completion mechanism.

Our empirical study in Section 4.9 using a session log encapsulating 6.1M sessions demonstrated that the proposed metrics show the best alignment with the user preferences exhibited in the session log.

The *pSaved* metric instantiated by the machine-learned prefix length-dependent examination probability function f_l^d achieves the weighted correlation level of 0.904 (Table 4.4) and correlation of 0.820 in an experiments simulating online A/B tests (Table 4.5). A close performance can be achieved by parameterising *pSaved* metric by the prefix-independent examination probability function f_l^i. We believe these two metrics can be recommended for the QAC evaluation.

Our results indicate that a higher alignment with the online evaluation metrics, such as success rate, can be achieved by using *pSaved* metrics. The highest alignment is achieved when the proposed metric is instantiated by the user behaviour models that are trained on the historical interaction data, thus supporting the statement of this thesis.

In turn, an improved alignment results in a higher agreement between offline and online steps of the evaluation pipeline (Section 3.1), thus increasing the overall effectiveness of the pipeline. This improvement forms the first step of the roadmap in Section 3.5. In the next chapter, we consider a complimentary approach for improving the evaluation pipeline efficiency. Specifically, assuming that the resource of the user interactions is limited, we propose to build a scheduler that prioritises experiments that are likely to be successful.

Chapter 5

Optimised Scheduling of Online Experiments

5.1 Introduction

In Chapter 4 we discussed how the efficiency of the evaluation pipeline can be improved by making offline evaluation better aligned with the online preferences of the users. Such improvements guarantee that the developed changes are less likely to be rejected in the online evaluation step and thus the overall efficiency of the evaluation pipeline is increased. In turn, this allows the search engine to evolve faster. Our work in Chapter 4 corresponds to the first point of the roadmap for improving the evaluation pipeline that we discussed in Section 3.5.

In this chapter, we proceed to the next point of this roadmap and our goal is to develop a scheduler that is capable of prioritising online experiments such that experiments that are likely to be successful are deployed earlier. This chapter is based on a publication (Kharitonov, Macdonald, Serdyukov & Ounis, 2015*b*).

Further, an experiment where the tested change (we denote the changed system as B) is shown to improve a considered metric with respect to the baseline system (denoted as A) is referred to as a *successful experiment*. A scheduler that prioritises potentially successful experiments will increase the efficiency of the evaluation pipeline. Indeed, since the number of experiments grows with the intensity of the search engine development (Kohavi *et al.*, 2013), 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 queue 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 could be processed, it is also beneficial to schedule only the promising experiments, without spending resources on the less promising ones.

To build the optimised scheduler, we re-use the historical interaction data in two ways. Firstly, we train the scheduler on the earlier deployed online experiments. Secondly, we use a click model that is trained on the historical interaction data as a predictor of the experiment's success. Hence, our work in this chapter supports the statement of this thesis (Section 1.3): we use historical interaction data to improve the evaluation pipeline efficiency.

In this chapter, we concentrate on the effectiveness-related experiments, where the changes that affect the ranking of the results are tested. Such experiments are the most numerous in the experimentation practice at Yandex, thus they form a class of experiments where scheduling can be useful. We start with describing the assumptions we make about the way online experiments are performed. We propose to reduce the problem of the optimal scheduling of such experiments to a learning-to-rank problem, where examples used for learning are formed from the historical interaction data, generated from experiments performed earlier. 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 chapter are three-fold:

- We formulate the 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;
- We evaluate the proposed scheduling algorithms over a large and representative dataset of online experiments.

The remainder of this chapter is organised as follows. After discussing the related work in Section 5.2, we review the assumptions we make to formalise the scheduling problem in Section 5.3. In Section 5.4 we discuss how the scheduling problem can be reduced to a learning-to-rank machine learning problem. In Section 5.5 we discuss the dataset we use in our evaluation study. The evaluation methodology is described in Section 5.6. The evaluation results we obtained are discussed in Section 5.7. We conclude this chapter in Section 5.8.

5.2 Related Work

Our work in this chapter relates to the existing research in improving the efficiency of online experimentation and shares the same goal: to make the online experimentation pipeline more efficient.

An approach to improve the scalability of the online evaluation was proposed by Tang *et al.* (2010), and we discussed it in Section 3.4. The central ideal of their approach is to allow each user interaction to participate in several online experiments, assuming that these experiments change independent variables (e.g. user interface and ranking). Even in the highly-scalable multi-layer evaluation framework proposed by Tang *et al.* (2010), in some layers (e.g. the ranking layer) we might want to run the most promising experiments earlier or the 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 an approach that is complementary to the one of Tang *et al.* (2010): 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 first.

An important step in our approach is to predict how likely a particular experiment is to be successful. Hofmann, Whiteson & de Rijke (2012) proposed to estimate the interleaving comparison outcomes by treating historical user sessions as comparison events between tested alternatives. In a recent work, Li *et al.* (2015) 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 chapter, we also use historical click data to predict the interleaving experiment outcome. However, there are considerable differences with the above discussed work (Hofmann, Whiteson & de Rijke, 2012; Li *et al.*, 2015). 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 & Craswell (2010) 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 in this chapter, 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.* (2012), who measured the correlation between $DCG@5$ and the absolute online metrics used in A/B tests. However, since the agreements reported in (Chapelle *et al.*, 2012; Radlinski & Craswell, 2010) are not perfect, the following question arises: Can a better prediction be achieved by using other features apart from the search result effectiveness? In our evaluation study in Section 5.7 we address this question. Further, Radlinski & Craswell (2010) performed their study on a dataset containing three experiments with major changes, and two experiments with minor improvements. However, major changes are likely to be rare in modern commercial search engines. In contrast,

we use a dataset, containing 82 real-life interleaving experiments, performed by a commercial search engine as part of its development. Thus, we argue that our study uses more a representative dataset.

As it can be seen from the above discussed literature, our work presented in this chapter finds a solid foundation in the earlier research and goes further in that we (a) state the problem of scheduling of online experiments (b) propose to use different offline and online predictors of experiment's success.

5.3 Scheduling Assumptions

We have discussed how the online evaluation is performed above in Section 2.3. In this section we state three assumptions about the way that the online experimentation queue operates, so that it becomes possible to formalise the scheduling problem. These assumptions specify the environment where our proposed scheduler operates. We specify 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 (Kohavi *et al.*, 2013). 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 (Kohavi *et al.*, 2013). This assumption is realistic and 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 (Leung, 2004)) until one of the two outcomes is achieved: one of the alternatives (A or B) wins the comparison, or the experiment's budget T is entirely exhausted. In some setups, experiments might stop before the budget is exhausted by applying a form of sequential testing (Chapter 7). In order to simplify the scheduling, we assume that such an early stopping scheme is not applied.

Under this assumption, 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 (Kohavi *et al.*, 2013). 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. Secondly, this assumption makes the whole online evaluation 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 chapter, 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 queue does not play an important role. We only require A1-A3 to hold. For instance, two experiments can run in 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.

Finally, we note that the described assumptions are very practical: we only assume that experiments typically have a similar number of interactions, are not interrupted, and should be scheduled according to how promising they are.

5.4 Optimising the Schedule

In this section, we firstly formulate the problem of optimised scheduling (Section 5.4.1), using the assumptions discussed in Section 5.3. Since this formulation relies on feature-based machine learning, in Section 5.4.2 we describe the feature representation of the online experiments we use. Finally, in Section 5.4.3 we describe our approach to reduce the scheduling problem to a learning-to-rank problem.

5.4.1 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 quantity can be informally considered to represent 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 then equal to

$$O = \sum_{i=1}^{|\mathbb{E}|} \left(P(S(i)) \mathbb{I} \left[\sum_{j=1}^{i} T_{S(j)} \leq T^* \right] \right) \tag{5.1}$$

where T^* is the total number of user interactions available for the experimentation, $T_{S(j)}$ 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 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:

$$S_0^{-1}(e_i) < S_0^{-1}(e_j) \Rightarrow P(e_i) \geq P(e_j) \tag{5.2}$$

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 (5.2) holds. Once a currently running experiment finishes, the firstly scheduled experiment ($S(1)$, with the highest value of $P(e)$) is deployed.

Under the assumptions A1-A3 this algorithm is optimal provided that the probabilities of experiment success $P(e)$ are available, i.e. it maximises Equation (5.1).

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, as the prioritisation of experiments is akin to the problem of ranking. Moreover, the learning-to-rank approach allows us to transparently use heterogeneous features to represent an experiment. Next, we discuss the feature representation (Section 5.4.2) and the machine learning algorithms (Section 5.4.3) used in our work.

5.4.2 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.

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 $Precision@n$, $ERR@n$ (Chapelle *et al.*, 2009), $DCG@n$ (Järvelin & Kekäläinen, 2002).

As the online metrics naturally reflect the relative importance of queries in the query stream, we weight the offline metrics according to the query frequencies. These frequencies we calculated using the same dataset that was used to calculate the click model-based features discussed below. This dataset contains interactions that preceded all of the experiments that we used in our evaluation study. The effectiveness-based features were calculated using the top 50 most frequent queries.

To get a effectiveness-based feature 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 non-relevant. 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 non-relevant, 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 intention. A possible way to address this is to use implicit feedback from the real users. In this chapter, 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) (Chapelle & Zhang, 2009) click model using a separate part of the dataset. We discussed this model above in Section 2.5. Under this model, for a fixed query, each document u has two

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

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
    $D \leftarrow D + \pi_i P_e(r) a_u (\mathbb{I}[r\ from\ A] - \mathbb{I}[r\ from\ B])$
    Probability of examining the next document
    $P_e(r+1) \leftarrow P_e(r)\,(1 - a_u s_u)$
  end
end

Algorithm 5.2: Calculating the expected difference in clicks obtained by A and B in an interleaving comparison.

parameters: 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 2.3 (Section 2.5).

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 5.1). 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 5.2).

In Algorithms 5.1 & 5.2 we use the notation introduced in Section 2.3.2: 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 π_i.

To calculate the expected difference in the number of clicks for a particular result page L_i, Algorithm 5.2 iterates over the results from top to bottom and maintains a variable $P_e(r)$ that equates to the probability of the user examining the result u at rank r. Assuming that the user examined the result u, they click on u with probability a_u and contribute a positive score if u belongs to the team A and negative otherwise. The expectation over the set of result pages $\mathbb{L}$ is achieved by weighting the per-page scores with the interleaving policy $\boldsymbol{\pi}$.

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 the outcome of this preliminary experiment (Section 2.3, Equation (2.7)). 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 pure exploration step in the ϵ-first greedy algorithms for the multi-armed bandit problems (Sutton & Barto, 1998; Tran-Thanh *et al.*, 2010). Indeed, given a set of experiments ("arms"), we need to identify which of them is more likely to be successful. However, we cannot use more advanced bandit algorithms, as interrupting experiments might have consequences we want to avoid (A2).

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 algorithm 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.

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.

Limitations The features we consider allow us to build a fine-grained representation of the candidate experiments. However, some of these features have a limited applicability. Indeed, effectiveness-based features assume that relevance labels can be provided, but that might be not always possible, e.g. in the case of personalised ranking. Further, the usefulness of the click model-based features is restricted to the experiments where historical click information is available, hence these features might be less useful in the experiments with mostly ranking of the tail (low frequency) queries changed. In contrast, the online exploration feature can be used universally across all types of the interleaving experiments.

5.4.3 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.

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 implemented in the scikit-learn[30] package (Pedregosa *et al.*, 2011). 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 (Ridgeway, 2004). The parameters (e.g. number of trees) are tuned through 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 experiment. $\mathbb{P}$ formulated as follows:

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

where Δ is defined in Equation (2.6) (Section 2.3.2, page 21).

[30] `http://scikit-learn.org`

Table 5.1: Descriptive statistics of the dataset.

#Exp	B wins A	Impressions: Min	Median	Mean	Max	Total
82	31	178K	1M	2M	39M	174M

We use the GBM (Ridgeway, 2004) package for R to find a function that minimises the number of misordered pairs. As an alternative pairwise ranker we use RankingSVM (Joachims, 2002).

5.5 Dataset

Before discussing the experimental study in the next section, we briefly describe the dataset used in this chapter. This dataset consists of the subset of the interleaving experiments performed by Yandex during a five week period in Spring 2014. It 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. On average, the interleaving experiments were deployed for 10 days.

Salient statistics of the dataset are provided in Table 5.1. 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 2.3.2 (page 22). The parameters of the sDBN model (Chapelle & Zhang, 2009) used for calculating the click model-based features (Section 5.4.2) are estimated using a separate query log sample from a two-week period. All experiments in the dataset were deployed after this period. All online experiments were deployed on the Russian market.

5.6 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:

RQ5.1 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?

RQ5.2 What are the best performing prediction algorithms? How do the proposed features compare to each other?

RQ5.3 How do the learned approaches compare in terms of the quality of the schedules they generate?

As we will discuss in Section 5.6.4, the random order we compare to in RQ5.1 is not an artificial baseline, instead it reflects the stochastic order of the experiments arriving to the experimentation queue 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 5.4.2, 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 separate features.

5.6.1 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, the 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 (Ling *et al.*, 2003).

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:

$$AUC(S) = \frac{\sum\limits_{e_1, e_2 \in \mathbb{R}} \mathbb{I}\left[\left(S^{-1}(e_1) - S^{-1}(e_2)\right)\left(\Delta(e_1) - \Delta(e_2)\right) < 0\right]}{|\mathbb{R}|} \tag{5.3}$$

5.6.2 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. However, 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 as budget T^*. Thus, we are interested in minimising the following measure, similar to Equation (5.1):

$$Q(S) = \sum_{i=1..|S|} \left(\mathbb{I}[B(e) \succ A(e)] \cdot \mathbb{I} \left[\sum_{j=1}^{i} T(e_i) \leq T^* \right] \right) \tag{5.4}$$

$Q(S)$ 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 $\left(\sum_{j=1}^{i} T(e_i) \leq T^*\right)$ 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 $T(e_i) = T = const$ is large enough for any experiment to have a definite outcome, then $Q(S) \approx Precision@R$, where $R = \frac{T^*}{n}$.

5.6.3 Statistical Methodology

To evaluate the quality of a schedule S, we perform a bootstrap estimation of the metric values $AUC(S)$ and $Q(S)$ using the dataset 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 T 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 sample the user interactions. After that, we proceed to the next scheduled experiment, $S(2)$. Again, we sample the user interactions. We proceed, until the limit T^* is reached, and calculate the value of $Q(S)$ according to Equation (5.4). 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.

Input: Number of user interactions available for an experiment T; 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 $train, test \leftarrow RandomStratifiedSplit(\mathbb{E}, nSplits = 10)$ **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 T 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:
      $\hat{T} = \hat{T} - T$; $j \leftarrow j + 1$
    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 5.3: The bootstrap-based evaluation protocol.

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 5.3.

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^*.

5.6.4 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 the work we presented in this chapter 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 $\Delta(e)$ of the experiment's outcome (Equation (2.6)). This baseline uses the data produced by the experiment, which is unavailable before the experiment was performed.

5.7 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 DCG3, ERR3, 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 experi-

Table 5.2: Performance of the scheduling algorithms, measured by AUC on the dataset of interleaving experiments. The values in bold are the highest in the corresponding row, excluding *UB*. The values denoted by $\triangle$ statistically significantly outperform other values in the same row (except for UB, $p < 0.05$).

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

ment. We include it as it resembles the interleaving experiment outcome prediction feature used further in Section 6.3 and thus it is interesting to compare to it.

We use the Wilcoxon test to compare the performance of the schedulers (excluding *UpperBound*).

5.7.1 Prediction Quality

In Table 5.2, we report the results of our evaluation of the quality of the experiment outcome prediction algorithms, measured on the dataset of the interleaving experiments. The quality is measured by AUC, and the measurement is performed by Algorithm 5.3. The values in denoted with $\triangle$ 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 dataset is deployed for 10^5 interactions. 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 Table 5.2, 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 5.2) 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 part of Table 5.2, #sample = 0), we observe that these schedulers (e.g. *LR* 0.76, *LGBM* 0.83, and *PGBM* 0.81) 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*). Next, 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, $p < 0.01$).

These observations allows us to answer RQ5.1: 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*.

As the number of the user interactions available for exploration grows, the performance of the exploration-based scheduler *Explore*: 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.

Interestingly, the machine-learned schedulers demonstrate a comparable performance when little or no exploration is used. Indeed, *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 5.2, 0.83). 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, 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 RQ5.2. *LGBM* outperforms other schedulers, and it improves over the best effectiveness-based scheduler *DCG* by a margin of 8%.

5.7.2 Evaluating the Schedule

In Table 5.3, we report our results obtained when evaluating the quality $Q(S)$ of the scheduling algorithms. 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[31] experiments ranked first for evaluation. The user interactions left after performing the exploration step are split for these three

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

Table 5.3: The quality of the scheduling algorithms measured by $Q(S)$ on the dataset of the interleaving experiments. The values denoted by $\triangle$ outperform other in the same scenario (T, #sample), $p < 0.05$ (except for UB). In bold are the highest metric values (except for UB) in the corresponding scenario.

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

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 un-scheduled (*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.

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 5.3). 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 5.3.

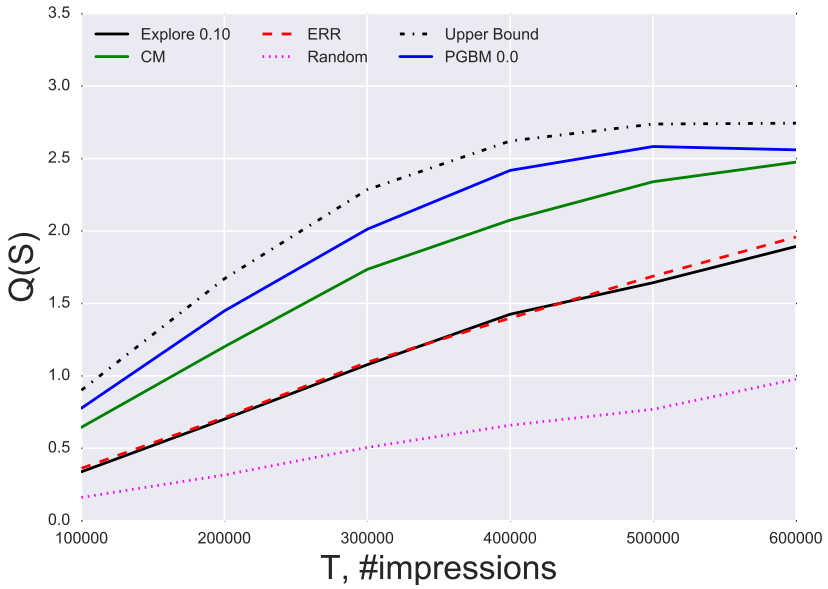
On comparing the results with T^* varied (left and right parts of Table 5.3, T^* is the total number of user interactions available for experimentation) we notice that as T^* increases, the values of the $Q(S)$ metric increase for all tested schedulers. In particular, the upper bound of the scheduling 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 some of the experiments "become" successful.

To obtain an additional insight into the behaviour of the schedulers, we vary the overall number of user interactions available to the experimentation queue (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 report the results in Figure 5.1. From Figure 5.1 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 RQ5.3. 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* 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.

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. 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.

Figure 5.1: Quality $Q(S)$ of the best schedulers as the number of the user sessions available grows, measured on the interleaving dataset.



5.8 Conclusions

In this chapter, we stated the problem of the optimal scheduling of the online experiments. In Section 5.3, we described three assumptions we make about the online experimentation 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.

Our findings suggest that our proposed machine-learned schedule optimisation algorithms outperforms the randomised, "natural" schedule by up to 342% (Table 5.3, $T^* = 4.5 \cdot 10^5$) when the number of the successful experiments performed under a limited number of available user interactions is measured. 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%, Table 5.3, $T^* = 4.5 \cdot 10^5$). Our study also suggests that an exploration-based scheduler

can achieve a considerable improvement over an unoptimised schedule (190% improvement, Table 5.3, $T^* = 4.5 \cdot 10^5$). Notably, this can be applied for experiments where neither relevance judgements, nor historical click data is available. Similarly, it can be used to schedule experiments in the query auto-completion experimentation layer.

Overall, in this chapter we discussed how to build an online experimentation queue scheduler. This work addresses the second point in our roadmap for increasing the efficiency of the evaluation pipeline in Section 3.5. Indeed, our proposed scheduler deploys the most promising experiments first, hence it ensures a better utilisation of the limited resource of the user interactions w.r.t. the number of obtained successful experiments. Thus the scheduling has a direct impact on the efficiency of the entire evaluation pipeline, as it allows the search engine to deploy more successful experiments per unit of time in contrast to the case when the experiments are sorted "naturally" in a random manner. Online experiments tend to last for several days at least (Chapelle *et al.*, 2012; Drutsa *et al.*, 2015; Kohavi *et al.*, 2013), hence the cost of deploying an unsuccessful experiment instead of a successful one is relatively high, as it delays the evolution of the search engine. By using our proposed scheduler, the frequency of such events can be reduced.

Our work in this chapter supports the statement of this thesis. Indeed, the historical interaction data is both used as the training data for our proposed scheduler (in the form of the already performed experiments), and is used to generate click model-based features, that are used by the scheduler to predict if an experiment is likely to be successful. Hence, our work in this chapter illustrates how historical interaction data can be re-used to improve the efficiency of the evaluation pipeline.

In this chapter we increased the efficiency of the evaluation pipeline by reducing the number of deployed unsuccessful experiments. Another approach for improving the evaluation pipeline efficiency is to reduce the duration of online experiments, both successful and not. This can be achieved by increasing the sensitivity of online evaluation methods and in the next chapter we discuss this approach.

Chapter 6

Improving Sensitivity of Interleaving Experiments

6.1 Introduction

In Chapter 4 we aimed to increase the efficiency of the evaluation pipeline (Chapter 3) by improving the alignment of the offline evaluation metrics with the online satisfaction indicators. After that, in Chapter 5, we discussed how to optimise the scheduling of online experiments so that online experiments that are likely to be successful are deployed first, and hence the efficiency of the evaluation pipeline is increased.

The online evaluation step is the most time-consuming step: a typical A/B experiment is deployed for a period of one or two weeks (Chapelle *et al.*, 2012; Drutsa *et al.*, 2015; Kohavi *et al.*, 2012, 2013; Schuth *et al.*, 2015), and a typical length of interleaving experiments reported in the literature is up to five days (Chapelle *et al.*, 2012; Schuth *et al.*, 2015). Consequently, even if we only deploy online experiments that are likely to be successful (e.g. by implementing our work in Chapter 4 and 5), it will still take days or weeks for each experiment to complete the evaluation, therefore considerably restricting the speed of search engine's development. It can be argued that improvements in the online evaluation efficiency would have the strongest impact on the overall evaluation efficiency. Motivated by this observation, in this and the following Chapters 7 & 8, we aim to increase the efficiency of the evaluation pipeline by reducing the duration of the online evaluation step.

When comparing two web document search ranking functions, interleaving was found to be faster to obtain the comparison outcome than an A/B test (Chapelle *et al.*, 2012; Schuth *et al.*, 2015). A variety of methods were proposed to further reduce the duration of interleaving experiments by improving the interleaving sensitivity.[32] Roughly, research in the existing literature can be divided into two areas:

[32] In Chapter 2, we defined *sensitivity* as ability to obtain a reliable comparison outcome with few observations.

optimisation of the click credit assignment (Chapelle *et al.*, 2012; Radlinski & Craswell, 2010; Yue *et al.*, 2010) (i.e. how important a particular click is); and optimisation of the probability of showing of interleaved result pages, or the interleaving policy (how often a particular interleaved result page is shown) (Radlinski & Craswell, 2013).

In this chapter, we discuss two approaches to improve the interleaving sensitivity. First, in Section 6.3, we discuss a click model-based approach to increase the interleaving sensitivity by means of the interleaving policy optimisation which was proposed in a publication (Kharitonov *et al.*, 2013*c*). However, such an approach has considerable limitations, and after discussing them, we propose to address these limitations by introducing the Generalised Team Draft interleaving framework. In contrast to the click model-based approach, Generalised Team Draft combines the policy optimisation with the click weight optimisation. Generalised Team Draft was introduced in (Kharitonov, Macdonald, Serdyukov & Ounis, 2015*a*). Both these approaches rely on re-using historical interaction data to optimise their parameters and increase interleaving sensitivity, hence they support the statement of this thesis (Section 1.3).

The Generalised Team Draft framework generalises the existing research in two aspects. First, we consider both the interleaving policy and the credit assignment function as parameters to optimise in our framework. As a result, our framework has a higher flexibility that can be used to achieve a higher sensitivity. Second, we formulate our framework to be general w.r.t. the actual presentation of the result pages, so that it can be applied for domains such as image search, where a grid-based presentation is used (see Figure 6.1 for an example). In contrast, the existing studies concentrate only on the document search domain.

An important property of an interleaving algorithm is its *unbiasedness*. Informally, we refer to an interleaving algorithm as biased, if its outcome is systematically shifter towards one the systems. Such a bias should be avoided as it introduces errors in the evaluation. As a part of our proposed Generalised Team Draft, we describe a formal criterion that interleaving policy and click feature representation must satisfy so that the interleaving evaluation is unbiased. This requirement can be applied in domains with the grid-based result representation.

In contrast, the current research in the interleaving policy optimisation (e.g. (Radlinski & Craswell, 2013)) explicitly relies on the list-based representation to formulate the unbiasedness criteria. Radlinski & Craswell (2013) consider a model of a randomly clicking user who uniformly selects the examination depth and then clicks on results among the examined. Such a formulation does not generalise to the grid-based result representation.

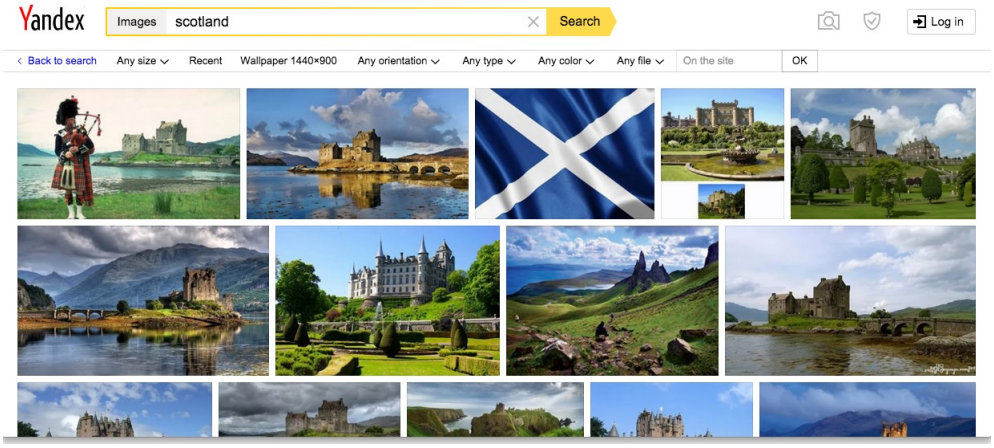


Figure 6.1: Image search result page as an example of the grid-based representation.

In this chapter, we introduce a stratified estimator of the interleaving experiment outcome which stratifies the interleaving interactions according to the teams of the results shown on the result pages. Under our proposed stratified estimator, the scores for each possible team combination are calculated independently and are combined after that. As a result of this procedure, the inter-strata variance is eliminated, thus increasing the interleaving sensitivity. In addition, this stratification simplifies the optimisation of the parameters performed by Generalised Team Draft.

By developing the Generalised Team Draft framework, we increase the evaluation pipeline efficiency from two perspectives. Firstly, interleaving comparisons are introduced in search domains with grid-based result representation, thus allowing faster evaluation of the system's effectiveness in domains such as image search. Secondly, our proposed framework improves the sensitivity of the interleaving methods both in domains with list- and grid-based representation, thus allowing interleaving experiments to be deployed on a less volume of the search traffic or for a smaller number of weeks. As we will further demonstrate in Chapter 8, when combined with improved statistical testing methods, Generalised Team Draft reduces the mean deployment time. Clearly, such a reduction will have a strong impact on the evaluation pipeline efficiency, as it reduces the time spent on its most time-consuming online evaluation step.

Overall, the contributions in this chapter are four-fold:

- We study a user model-based approach to increase the interleaving sensitivity;
- Next, we propose a principled, data-driven Generalised Team Draft framework that addresses weaknesses of our previous approach and overcomes the limitations of the approaches proposed

in the literature. Generalised Team Draft achieves sensitive interleaving by combining the stratification, the interleaving policy optimisation, and the credit function learning. Moreover, it can be applied in domains with the list-based and the grid-based result presentations;

- We propose sufficient conditions that the click feature representation and the interleaving policy need to satisfy so that the Generalised Team Draft interleaving remains unbiased;
- We perform a large-scale evaluation study of Generalised Team Draft, using two datasets that contain document and image search online experiments.

The remainder of this chapter is organised as follows. In Section 6.2 we discuss the related work. In Section 6.3 we describe our initial approach to improving interleaving sensitivity and discuss its limitations. Further, in Section 6.4 we define the Generalised Team Draft interleaving framework and discuss some of its aspects in Section 6.5. Our proposed stratification technique, and how to optimise the interleaving parameters, is discussed in Sections 6.6 and 6.7, respectively. We discuss the datasets we use in our experimental study in Section 6.8. In Section 6.9 we describe the instantiations of our framework for the web document search and for the image search domains. The evaluation scenario we use are described in Section 6.10. We discuss our obtained results in Section 6.11. We conclude this chapter in Section 6.12.

6.2 Related Work

Since the introduction of the first interleaving method, Balanced Interleaving (Joachims, 2002, 2003), several other interleaving methods were proposed, including Team Draft (Radlinski *et al.*, 2008), Probabilistic Interleaving (Hofmann *et al.*, 2011), and Optimised Interleaving (Radlinski & Craswell, 2013). An important characteristic of an interleaving method is its sensitivity, i.e. its ability to obtain a reliable experiment outcome with as few user interactions as possible. The problem of increasing the sensitivity of an interleaving method has attracted a considerable attention from the research community, and below we review the most relevant work in this area.

We split this review in two parts: in Section 6.2.1 we discuss the approaches proposed to improve the interleaving sensitivity by optimising the click aggregation schemes, and in Section 6.2.2 we discuss how a higher sensitivity can be achieved by controlling the interleaving policy.

6.2.1 Click Score Optimisation

Yue *et al.* (2010) proposed a method to learn a more sensitive credit assignment function for the Team Draft interleaving experiments. Later, this approach was also discussed by Chapelle *et al.* (2012). Informally, the core idea of Yue *et al.* (2010) is to learn how to weight user clicks in the interleaving comparisons so that the confidence in the already performed experiments is maximised. As a result, new interleaving experiments will achieve the required level of confidence in their outcomes with fewer user interactions, i.e. the interleaving method will have a higher sensitivity. Yue et al. refer to this learning problem as an "inverse" hypothesis test: given user interaction data for the comparisons with known outcomes, one learns a credit assignment function that maximises the power of the test statistic in these comparisons.

Our work on Generalised Team Draft in this chapter is based on the ideas of Yue *et al.* (2010), and aims to overcome two shortcomings of their approach. First, it is not straightforwardly clear what kind of features and weighting functions are allowed so that no biases are introduced when learning the credit assignment function. It is possible to build an example of the click feature representation that makes the credit function learning process prone to biases (Section 6.5). In our work, we propose a formal unbiasedness requirement that ensures that a feature-based credit assignment function is not biased. Moreover, we propose a restricted family of the click features that allows us to make this requirement easy to operate in practice.

Second, Yue *et al.* (2010) assume that the interleaving policy (the probabilities of showing the different interleaved result pages) is fixed. In this chapter we consider the interleaving policy as an optimised parameter. In the user model-based interleaving optimisation approach, the interleaving policy is optimised w.r.t. a user model trained on the historical interaction data. In Generalised Team Draft, we optimise both the interleaving policy and the credit assignment function directly on the historical interaction data, recorded in the experiments.

One of the approaches to improve the interleaving sensitivity we discuss is stratification, a simple yet effective technique that has its roots in the Monte-Carlo stratified sampling methods (Asmussen & Glynn, 2007; Robert & Casella, 2009). Its application for online A/B tests was studied in (Deng *et al.*, 2013), but has not previously been considered in the context of interleaving.

6.2.2 Interleaving Policy Optimisation

Radlinski & Craswell (2013) proposed the Optimised Interleaving framework, which specifies a set of requirements that an interleaving method has to meet so that (a) its results are not biased, (b) it is sensitive, and (c) the users are not too frustrated by the poor relevance of the top results in the interleaved re-

sults pages. However, Optimised Interleaving has some limitations, which we aim to overcome. Firstly, it is formulated specifically with a particular web document search user model in mind, and the interleaving policy optimisation it performs is formulated with respect to a click model that is specific for the list-based result presentation. This hinders extending the interleaving approaches to other domains.

Furthermore, the unbiasedness criterion proposed by Radlinski & Craswell (2013) uses a model of a randomly clicking user that makes it hard to combine with click weighting schemes that depend on click features. As an illustration, consider a scenario where the score of a click is a linear combination of its dwell time and the position of the clicked result. In order to apply the unbiasedness criterion used in Optimised Interleaving, one would need to specify a model of a randomly clicking user that has a realistic joint distribution of the clicked positions and dwell times. As the number of features used to represent a click grows, specifying such a distribution might become a prohibitive task. Finally, in Optimised Interleaving, the interleaving policy is optimised in a data-free manner.

In our study of the user model-based interleaving policy optimisation in Section 6.3, we address the last limitation. Specifically, we argue that an improved sensitivity can be obtained if a data-centric optimisation of interleaving policy is performed. However, this approach is still limited by the document search domain and simple scoring schemes, and requires a sophisticated run-time system.

Thus, in Section 6.4 we further propose the Generalised Team Draft framework, which is built upon Optimised Interleaving, but addresses all the discussed limitations. Firstly, it specifies a generalised unbiasedness requirement that can be applied for the grid-based result pages. Second, Generalised Team Draft performs a joint data-driven optimisation of the interleaving policy and the credit assignment function. Further, its performance can be evaluated using a set of historical experiments, available to each search engine that uses Team Draft-based interleaving experiments, in contrast to Optimised Interleaving, which uses different result list pages. Finally, Generalised Team Draft relies on the actual distribution of the click features observed in the historical interaction data, thus no modelling of the joint distribution of the click features is required.

Chuklin, Schuth, Hofmann, Serdyukov & de Rijke (2013) proposed an interleaving method that goes beyond the classic "ten blue links" web document presentation and deals with the vertical results (e.g. News, Images, Finance) incorporated in the main web search result page. However, the challenges that Chuklin et al. address (e.g., ensuring that in the interleaved result page the vertical results are still grouped) are quite different from the problems faced when developing an interleaving mechanism for a new domain, e.g. image search. In the latter case, one needs to decide how to specify the credit assignment function, or how to select the interleaving policy. To the best of our knowledge, our work is the first to address the problem of interleaving in a domain with the grid-based result presentation.

Overall, our Generalised Team Draft framework finds a solid foundation in the research discussed above, but it also addresses several shortcomings of the earlier approaches (Radlinski & Craswell, 2013; Yue *et al.*, 2010). We introduce our framework in Section 6.4. However, before that, we discuss how a user model-based approach can be used to increase the interleaving sensitivity.

6.3 User Model-based Sensitivity Optimisation

In Section 2.3.2 (page 20) we defined an interleaving policy $\boldsymbol{\pi}$ as a vector with its components π_i defining how often a particular combination of teams L_i is demonstrated to the users. In this section, we hypothesise that organising the interleaving policy in such a way that the combinations L_i that are unlikely to contribute to the interleaving score (Equation (2.6), page 21) are shown as rarely as possible, should improve the ability to derive reliable conclusions with less impressions. In other words, with the total number of user interactions in an interleaved experiment being fixed, the interleaved result pages that often lead to ties[33] should be shown less frequently in comparison with those that witness a contrast between A and B.

In Section 6.3.1, we study how this goal can be achieved based on the Optimised Interleaving framework proposed by Radlinski & Craswell (2013), initially assuming that information about the future user behaviour is available at the start of the experiment. We will relax this assumption in Section 6.3.2 by proposing a user model-based approach to calculate estimates of the used statistics.

6.3.1 Optimisation

As introduced in Section 2.3.2, the outcome of an interleaving experiment is represented as the mean score over all interactions:

$$\Delta = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} S(a) \tag{6.1}$$

where $\mathbb{A}$ is the set of the user interactions in the experiment and $S(a)$ is the score assigned to an individual interaction a.

Using definitions from Section 2.3.2 and denoting the set of interactions with the ith team combination demonstrated to the users as $\mathbb{A}_i$, we can re-write Equation (6.1) as follows:

$$\Delta = \frac{1}{|\mathbb{A}|} \sum_i \sum_{a \in \mathbb{A}_i} S(a) = \sum_i \frac{\sum_{a \in \mathbb{A}_i} S(a)}{|\mathbb{A}|} = \sum_i \pi_i \cdot \bar{S}_i \tag{6.2}$$

[33] For instance, the result pages that do not attract clicks or often have similar credits assigned to both alternatives.

where $\bar{S}_i$ is the mean score obtained by interactions with the ith team combination:

$$\bar{S}_i = \frac{1}{|\mathbb{A}_i|} \sum_{a \in \mathbb{A}_i} S(a) \tag{6.3}$$

Intuitively, with the total number of impressions $|\mathbb{A}|$ fixed, higher absolute values of Δ correspond to higher contrast between A and B, and lead to the ability to determine the experiment outcome with higher reliability. Thus, let us consider a simple upper bound on the absolute value of Δ:

$$|\Delta| = \left| \sum_i \pi_i \bar{S}_i \right| \leq \sum_i \pi_i \left| \bar{S}_i \right| \tag{6.4}$$

Indeed, the absolute value of Δ is bounded by the product of the experiment policy and the statistics of the interleaved results lists $|S_i|$. This quantity is related to the contribution that the impressions with the team combination L_i make to the difference (6.1), as we discussed earlier. This bound also provides us with an idea how the experiment policy can be adjusted before starting the experiment. Despite the fact that a higher upper bound does not necessary imply a higher value of Δ, in this section we argue that controlling π_i, so that the upper bound increases, leads to higher sensitivity of the interleaving. Intuitively, this idea can be expressed as follows: with everything else being equal, it is generally better not to show a result list L_i with a low value of $|\bar{S}_i|$. In the following, we assume that for each query we can predict a vector $\boldsymbol{\mu}$, such that its components approximate $\bar{S}_i$, i.e. $\mu_i \approx |\bar{S}_i|$. How to obtain such a vector is discussed in Section 6.3.2.

As discussed above, our goal is to adjust the policy $\boldsymbol{\pi}$ so that the upper bound (6.4) increases. In order to achieve that, we leverage the Optimised Interleaving framework (Radlinski & Craswell, 2013), which can be used to optimise the interleaving experiment properties without introducing biases. In particular, Radlinski & Craswell (2013) define a criterion of the unbiased interleaving policy: the expected credit from a randomly clicking user should be zero.

To formalise this requirement, we assume that the score of the interaction a is obtained under the deduped binary scheme (Section 2.3.2), i.e. it is equal to the sign of the difference between the number of clicks c on results from team B and A. This scheme assumes that clicks on top d positions are ignored if the top d results in A and B coincide.

For the sake of convenience, we introduce an indicator function $T_i(p)$, that equates to $+1$ (-1) if a result on position p of the result page L_i comes from the team B (A):

$$T_i(p) = -\mathbb{I}\left[L_i(p) \in TeamA\right] + \mathbb{I}\left[L_i(p) \in TeamB\right] \tag{6.5}$$

Further, by $p(c)$ we denote the position of the click c. Then, the score $S(a)$ of an interation a with an interleaved result page i demonstrated to the user, can be found as follows:

$$S(a) = sign\left(\sum_{c \in a} T_i(p(c))\right) \tag{6.6}$$

Using this notation, we can formalise the unbiasedness requirement (Radlinski & Craswell, 2013) that the set of interleaved result lists $\mathbb{L}$ and the interleaving policy $\boldsymbol{\pi}$ have to meet:

$$\forall k \sum_{i=1}^{|\mathbb{L}|} \pi_i \cdot sign\left(\sum_{p=1}^{k} T_i(p)\right) = 0 \tag{6.7}$$

Assuming that $\boldsymbol{\mu}$ is known and combining Equations (6.4) and (6.7) we formulate our optimisation problem as follows:

$$\boldsymbol{\mu}^T \boldsymbol{\pi} \rightarrow \max \tag{6.8a}$$

$$\forall k \sum_{i=1}^{|L|} \pi_i \cdot sign\left(\sum_{p=1}^{k} T_i(p)\right) = 0 \tag{6.8b}$$

$$\sum_i \pi_i = 1 \tag{6.8c}$$

$$\forall i \; \pi_i \geq 0 \tag{6.8d}$$

Indeed, the solution of the optimisation problem stated by the set of Equations (6.8) maximises the upper bound of the experiment's outcome $|\Delta|$ (6.8a), meets the unbiasedness condition (6.8b) proposed by Radlinski & Craswell (2013), and represents a valid distribution (6.8c & 6.8d). We argue that setting the interleaving experiment policy to the solution of the linear optimisation problem (6.8) leads to a higher sensitivity of the experiment.

Since the components of $\boldsymbol{\mu}$ are predicted, this can cause undesired noise to the solution. Indeed, a small variation in μ_i might result in the linear programming problem (6.8) having a completely different solution. In order to reduce this noise, we introduce a regularisation term to the optimisation objective (6.8a) that adds a penalty to solutions that diverge too far from the uniform policy $\boldsymbol{\pi}_U$.[34] Thus, we replace objective (6.8a) with the following expression:

$$\boldsymbol{\mu}^T \boldsymbol{\pi} - \alpha(\boldsymbol{\pi} - \boldsymbol{\pi}_U)^T(\boldsymbol{\pi} - \boldsymbol{\pi}_U) \rightarrow \max \tag{6.8a*}$$

where α is a non-negative scalar parameter. With α being zero, (6.8a*) reduces to (6.8a), while large values of α force the solution to be the uniform vector. In the latter case, the policy optimisation

[34] In the experimental part of this paper, we work with the Team Draft-based set of interleaved result lists $\mathbb{L}$, so $\boldsymbol{\pi}_U$ is a feasible solution of the optimisation problem (6.8).

is ignored and all team combinations are demonstrated to the users with equal probabilities and the solution of (6.8) becomes the Team Draft interleaving with the deduped binary credit scheme. On the other hand, with $\alpha = 0$ the optimal solution of (6.8) is completely defined by possibly noisy estimates of $\boldsymbol{\mu}$. In this section, we consider α to be the same for all queries. However, it can be used to controlled to reflect our confidence in the quality of predictions for each query. In the next section, we discuss how the estimates of $\boldsymbol{\mu}$ are obtained.

6.3.2 Using the Historical Interaction Data

The general idea behind predicting the parameters $\boldsymbol{\mu}$ of the optimisation problem (6.8) is the following. Having observed a massive click log representing the users' behaviour, we can train a generative model of the user click patterns. Once the click model with the pre-trained parameters is available, it can be used to model how users will behave once the result page is modified. A variety of generative click models have been proposed so far, (Chapelle & Zhang, 2009; Chuklin *et al.*, 2015; Craswell *et al.*, 2008; Dupret & Piwowarski, 2008; Guo *et al.*, 2009). In this section, we use a simple yet effective modification of the Dynamic Bayesian Network, Simplified DBN (sDBN)[35], proposed by Chapelle & Zhang (2009). We have discussed this model in Section 2.5 and applied it to predict the user's clicking behaviour in Chapter 5.

Recall that the sDBN model assumes that a user examines the result list from top to bottom. After examining a document u, the user either finds it attractive with probability a_u and clicks on it, or continues to the next document. After clicking on a document, the user is satisfied with probability s_u and stops the examination process. Thus, for a fixed query, the model has two parameters per document u: attractiveness a_u and the probability of satisfying the user s_u. These parameters are learned from a click log by means of Algorithm 2.3 (Section 2.5, page 29). This learning procedure imposes Beta priors on the model parameters and following Chapelle & Zhang (2009) we set them to 1, i.e. $\alpha_a = \alpha_s = \beta_a = \beta_s = 1$.

After training the model parameters it can be used to predict $\boldsymbol{\mu}$. According to our definition, μ_i equates to the absolute value of the mean score obtained for a particular combination of teams on a result page. Under the sDBN model, this quantity can be calculated by means of Algorithm 6.1.

6.3.3 Qualitative Study

In the following, we use an offline evaluation approach to compare our proposed user model-based sensitivity to the standard Team Draft algorithm. Our study uses two datasets obtained from Yandex.

[35] We have also tried the Dependent Click Model Guo *et al.* (2009) and found it to perform worse.

Input: Parameters of the click model, a_u, s_u; set of interleaved result lists, L
Output: Vector of the optimisation objective (6.8a*) parameters $\boldsymbol{\mu}$
//$P_e(p)$ denotes the probability of examining the position p
$P_e(1) \leftarrow 1$
foreach $L_i \in \mathbb{L}$ **do**
  for $p \leftarrow 1$ *to* $|L_i|$ **do**
    $u \leftarrow L_i(p)$ // u is the document on the position p
    Expected credit from the position p
    $\mu_i \leftarrow \mu_i + P_e(p)T_i(p)a_u$
    Probability of examining the next document
    $P_e(p+1) \leftarrow P_e(p)\,(1 - a_u s_u)$
  end
  $\mu_i \leftarrow |\mu_i|$
end

Algorithm 6.1: Estimating $\boldsymbol{\mu}$ with the pre-trained sDBN click model.

The first datasets represents the pre-experimental user behaviour and is used to train the click model parameters. It is further referred to as the *user modelling* dataset. The second dataset contains six interleaving experiments. We refer to it as to the *experimental* dataset. In order to simulate a real-life scenario, the datasets are sampled over consequent non-overlapping time periods, with the user modelling dataset preceding the experimental dataset.

In order to collect the user modelling dataset, we apply the following filtering: firstly, we exclude all sessions that are affected by any online experiment, as that might result into a bias in the evaluation of the ability of the click model to predict the parameter μ; we remove all sessions with no documents clicked, as well as sessions with more than ten results examined since those can introduce an additional noise to the sDBN model. In order to balance the dataset size, the freshness of the click model parameters and the dataset sparseness, we use the following strategy: for the top 100 most frequent queries we collect the users' click behaviour over a period of week; for the rest of the queries we collect user behaviour data from the eight previous weeks as well. All queries are normalised to lowercase. The same user modelling dataset is used in all experiments.

The interleaving experiments are sampled from the query log, starting from the day after the click modelling dataset time span ended. The experimental data represents the users' click behaviour recorded while performing six ($E_1 ... E_6$) Team Draft-based interleaving experiments in April-May, 2013. In order to reduce variance in the offline evaluation, for a particular interleaving experiment, we keep only those queries that have every possible combination of teams shown to the users at least once.

Next, in order to avoid sparsity, we consider only the top six results in each result list. To further reduce noise, we exclude queries with results that are examined less than two times in the user modelling dataset. Queries with equal result lists for both A and B are removed as non-informative under the

Table 6.1: Experimental datasets statistics.

Name	#queries	#impressions	#CM sessions	winner
E_1	1,311	181,981	3,682,895	A
E_2	524	82,008	2,771,280	B
E_3	468	119,287	3,198,372	A
E_4	1,502	109,596	5,691,939	A
E_5	1,255	52,314	2,045,122	A
E_6	279	9,175	1,100,874	B

deduped credit assignment. The total number of user sessions used to train the click model parameters for queries in the experimental dataset is referred to as the click modelling sessions (CM sessions). We present the statistics on the whole dataset in Table 6.1. In the experiments considered, the A ranking function represents the production search system, while B corresponds to the experimental ranking. Winners are defined as a result of Team Draft on the considered interleaving dataset (bootstrap test, $p \leq 0.05$).

In this section, we use bootstrap sampling to estimate the probability of obtaining the correct (ground-truth) experiment outcome provided a fixed number of user impressions sampled (Chapelle *et al.*, 2012). More specifically, we vary the number of interaction sampled. On each step, we sample with replacement a specified number of interactions from the experimental data 1000 times, so that the distribution of the team combinations specified by the optimised interleaving policy equates to $\boldsymbol{\pi}$. In turn, $\boldsymbol{\pi}$ is determined by solving the optimisation problem (6.8). The frequency of the errors under such a sampling procedure is related to the level of confidence under the bootstrap test. A higher confidence with the same amount of observations indicate a higher level of sensitivity.

A visual representation of the bootstrapping sampling outcomes is presented in Figure 6.2. The plots correspond to experiments from E_1 to E_6, and each plot represents the probability of obtaining an incorrect experiment outcome after considering a particular number of impressions. We present results that correspond to Team Draft (i.e. $\alpha \to \infty$) and the solution of the optimisation problem (6.8) with $\alpha \in \{0.0, 0.5\}$. The deduped credit aggregation scheme is used. We notice that the proposed algorithm outperforms Team Draft in most of the experiments, and by varying α it is possible to control the optimisation risk: with $\alpha = 0$ the proposed algorithm demonstrates high sensitivity gains in E_3, E_4, E_5, E_6 but underperforms in E_2 with respect to Team Draft. On the contrary, with $\alpha = 0.5$ the maximum loss is almost negligible (E_1) while there are still significant improvements in E_3, E_4, E_5, E_6 and a slight improvement in E_2.

Overall, these results suggest that the historical user behaviour information can be used to improve

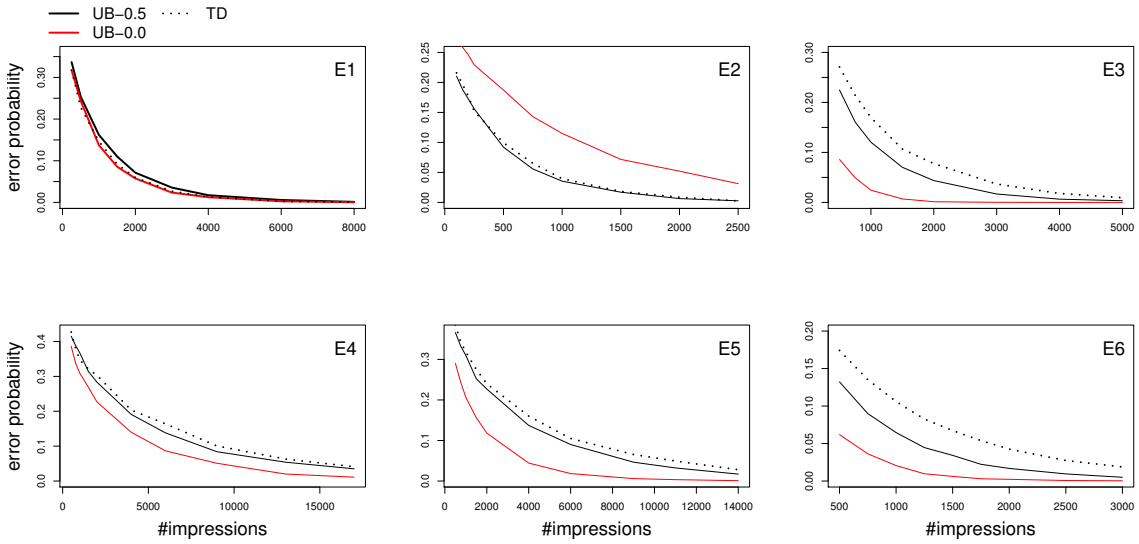


Figure 6.2: Comparison of the interleaving methods sensitivity. TD denotes Team Draft, UB-0.0 and UB-0.5 correspond to interleaving optimised with respect to historical user behaviour with α equal to 0.0 and 0.5, respectively. The deduped binary credit aggregation scheme is considered.

the sensitivity of the interleaving algorithms. Our small-scale experiments demonstrated that the proposed interleaving algorithm, which adjusts the interleaving policy according to the solution of the optimisation problem (6.8), has generally a higher sensitivity than Team Draft. Moreover, by controlling the algorithm's regularisation parameter α we can control the variance in the sensitivity gains.

However, the user model-based approach discussed in this section has considerable limitations. In the next section we discuss these limitations and in Section 6.4 we introduce the Generalised Team Draft framework which can overcome these limitations.

6.3.4 Discussion and Limitations

From our qualitative study in Section 6.3.3, we observe that our proposed user model-based approach for the interleaving sensitivity optimisation supports the statement of this thesis (Section 1.3). Indeed, by using the sDBN click model trained on historical interaction data, we are capable of increasing interleaving sensitivity, thus improving the efficiency of the whole evaluation pipeline (Chapter 3), as it allows interleaving experiments to be deployed for a shorter time. This direction of work corresponds to the third point of our proposed pipeline for improving the evaluation pipeline (Section 3.5).

Some limitations of the approach discussed in Section 6.3 come from the fact that it relies on the user model-based modelling. Indeed, in order to determine the optimal interleaving policy, the optimisation

problem (6.8) needs to be solved for each submitted query. This implies that (a) the parameters $\boldsymbol{\mu}$ need to be calculated in runtime (b) thus, click model parameters a_u and s_u need to be stored and accessed in run-time. Moreover, due to the heavy-tail character of the query distribution, the caching of results might be not effective.

Because of the use of the click models, we restricted the sensitivity optimisation approach to the domains where generative click models were proposed. At the same time, very few models were discussed in domains where results are not mere links arranged in a ranking list, such as image or video search.

Another problem is that the click model parameters can be estimated only for the queries that were submitted earlier. Consequently, the optimisation problem (6.8) makes only sense for queries with available click data. Thus the optimisation can be performed only for relatively frequent, head queries and necessarily the interleaving for the remaining queries has to be performed by the baseline Team Draft algorithm. As a result, the sensitivity of interleaving might be uneven for top frequent and long-tail queries, resulting in across-query biases.

Finally, since we rely on the existing click models, we are restricted in the way a click can be represented. Indeed, assume we want to alter the weight of the click depending on its dwell time or the user action after the click. Under the above discussed click model-based approach, to calculate the parameter $\boldsymbol{\mu}$ (Section 6.3.1) and specify the unbiasedness requirement (6.8b), we need to use a click model that realistically models the joint distribution of the click parameters (e.g. dwell time and the user actions on the clicked page). This constraint is very restrictive, as most of the existing click models only predict the fact of click, but not its parameters.

A possible approach to address that is to leverage the user behaviour data directly, without relying on the trained click models. In the remainder of this chapter we discuss the Generalised Team Draft interleaving framework that uses this direct approach. We argue and show that it addresses all the aforementioned concerns and, at the same time, achieves an increased interleaving sensitivity.

6.4 Generalised Team Draft Interleaving

The works of Yue *et al.* (2010) and Radlinski & Craswell (2013) on sensitivity optimisation based on credit assignment and policy optimisation, discussed in Section 6.2, lay the foundation for our Generalised Team Draft framework. However, our framework has significant differences from these works in that our proposed framework performs a joint optimisation of the interleaving policy and the credit assignment function, while Yue et al. and Radlinski and Craswell optimise only one of these parameters. Further, our framework can be applied for search domains with grid-based result pages. Below, we

introduce a formal requirement that a feature-based credit assignment function, the click feature representation, and the interleaving policy have to meet for the interleaving to be unbiased, regardless of the actual result representation. In contrast, Yue et al. do not discuss possible biases that can emerge due to feature-based learning, and Radlinski and Craswell only discuss feature-less credit assignment rules. By addressing the above discussed gaps, we build a sensitive interleaving framework that generalises the approaches proposed by Yue *et al.* (2010) and Radlinski & Craswell (2013).

Generalised Team Draft optimises the interleaving parameters on the level of the entire experiments, with their query distributions being representative of the whole query stream. As a result, the possibility of arising any inter-query bias is low, in contrast to the case of the query-level optimisation we discussed in Section 6.3.

In our framework, we consider the result pages that are obtained by applying the Team Draft mixing algorithm to the lists of the results of the underlying rankers A and B, sorted according to their relevance. The Team Draft mixing algorithm was proposed by Radlinski *et al.* (2008) and is described in Algorithm 2.2 (page 21) in Chapter 2. The exact mapping of the sorted result list into a result page is domain-specific (the list-based for document search, or the grid-based for image search, illustrated in Figure 6.3). Assuming that under this mapping the results ranked higher in the ranked list are mapped into positions with higher examination probability, mixing the sorted result lists of the rankers A and B according to Team Draft will result in a result page that cannot be more frustrating for the users than both the result pages generated from outputs of A and B. Due to this assumption we avoid the necessity of specifying the mixing algorithm for each possible domain-specific presentation, and can work with the underlying ranker output, which is always list-wise in practice. Apart from that, relying on the Team Draft-based result pages allows us to re-use a large-scale dataset of the experiments collected by an existing search engine for our evaluation study (Section 6.11).

Recall that the Team Draft mixing algorithm (Section 2.3.2, Algorithm 2.2, page 21) builds the interleaved result list in steps. At each step, both teams A and B contribute one result each to the combined list. Each team contributes the result that it ranks highest among those that are not in the combined list. However, the team that contributes first at each step is decided by a coin toss. For instance, as there are usually 10 results on a document search result page, 5 coin tosses are required to build it. Thus, there are exactly $2^5 = 32$ different combinations of the result teams on a result page[36].

We define Generalised Team Draft by stating four framework components (F1-F4), two requirements the components must satisfy (R1 and R2), and an optimisation objective that is used to adjust the framework components (O1).

[36]They can be enumerated as *abababab*, *ababababba*, *abababbaab*, ..., *babababaa*.

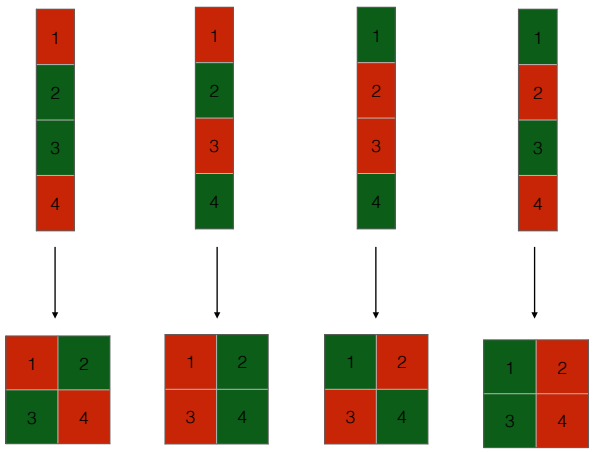


Figure 6.3: A possible left-to-right/top-to-bottom mapping illustrated on four interleaved ranked lists. This mapping maps the ranked lists to their corresponding 2D grid-based domain-specific representations. This mapping is used in our experiments with image search.

Now we can define the first component of our Generalised Team Draft framework:

F1 The set $\{(L_i, T_i)\}_{i=1}^{l}$ of the pairs of the interleaved result pages $L_i \in \mathbb{L}$ and their corresponding combinations of the result teams $T_i \in \mathbb{T}$. The result pages $\mathbb{L}$ are obtained by applying Algorithm 2.2 (Section 2.3.2, page 21) to the sorted outputs of the rankers A and B. Further, these interleaved ranked lists are mapped to their domain-specific representation. As in Equation (6.5), we define $T_i(p)$ to be equal to 1 (-1) if the team of the result on position p of the interleaved list that produced L_i is B (A);

It is possible that some pairs (L_i, T_i) contain identical result pages L_i, despite that the team combinations T_i associated with them are different (e.g. if A and B produce identical result lists). We consider such pairs to be different.

Following Radlinski & Craswell (2013), we explicitly define the interleaving policy as a parameter of the framework:

F2 An interleaving policy $\boldsymbol{\pi} \in \mathbb{R}^l$ is a vector that determines the probability of using a particular team combination when building an interleaved result page: $\pi_i = P(T_i)$;

Under our framework, the interleaving policy is the same for all queries and interleaving experiments. Informally, it can be considered as a distribution over the random seeds that can be used to "initialise" the coin used in Team Draft (Algorithm 2.2, page 21).

From (Yue *et al.*, 2010) we adopt the feature representation of the user's click $\phi(\cdot)$ and the form of the credit assignment function S:

F3 A function $\boldsymbol{\phi}(\cdot)$ that maps a user click c on an interleaved result page to its feature vector representation $\boldsymbol{\phi}(c) \in \mathbb{R}^n$. We also define an auxiliary indicator $T(c)$ that equates to 1 (-1) if the team of the clicked result is B (A);

F4 A scoring rule, $S = S(a; \boldsymbol{w}) = \sum_{c \in a} T(c) \cdot \boldsymbol{w}^T \boldsymbol{\phi}(c)$ that maps a sequence of clicks in the interaction a to the score of the alternative B. The vector $\boldsymbol{w}$ is a parameter, $\boldsymbol{w} \in \mathbb{R}^n$.

After running an experiment e, the score statistic $\Delta(e)$ can be calculated:

$$\Delta(e) = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} S(a; \boldsymbol{w}) \tag{6.11}$$

where $\mathbb{A}$ is a set of the user interactions in the experiment e. If $\Delta(e)$ is statistically significantly above zero, it is concluded that B outperforms A in the experiment e, as discussed in Chapter 2 (Section 2.3).

To ensure that the interleaving is unbiased, Radlinski & Craswell (2013) suggested the following criterion for the document search scenario: a randomly clicking user should not create any preference between A and B. To formalise this idea, they considered a user who (a) samples the number of the considered top results k randomly and (b) clicks uniformly at random on η results from the top-k results. This formulation explicitly relies on a list-based presentation. Furthermore, in our case the formalisation is even more challenging as the credit $S(a; \boldsymbol{w})$ is a function itself, since some feature representations might be prone to biases (we discuss this further in Section 6.5). We propose the following generalisation of the unbiasedness criterion from (Radlinski & Craswell, 2013):

R1 For any fixed sequence of clicks, the expectation of the total credit over the all pairs (L_i, T_i) of the interleaved pages L_i and distributions of teams T_i should be zero. Denoting the length of the sequence as J, the positions clicked as $p_1, p_2, ..., p_J$, and their corresponding click features as $\phi_1, \phi_2, ..., \phi_J$ we formalise this requirement as follows:

$$\forall J,\ \forall \{(p_j, \phi_j)\}_{j=1}^{J}\ \sum_i \pi_i \cdot \sum_j T_i(p_j) \cdot \boldsymbol{w}^T \phi_j = 0$$

Due to the linearity of the expectation, R1 is sufficient to guarantee the absence of the preferences for any randomised combination of the click sequences. Informally, this guarantees that a user who specifies an arbitrary interaction scenario that does not depend on the presented documents (e.g., "click on the first position, sample the dwell time uniformly from [0, 30], click on the third result, ...") will not create any preference for A or B in expectation.

Next, we require the policy $\boldsymbol{\pi}$ to be a valid distribution:

R2 $\forall i \;\; \pi_i \geq 0; \;\; \sum_i \pi_i = 1$

Among all of the possible combinations of $\{\boldsymbol{\pi}, \boldsymbol{w}\}$ that satisfy R1 and R2, we want to select the combination that maximises the interleaving sensitivity. Based on (Yue *et al.*, 2010), we use a *dissimilarity* measure D between the compared alternatives in a set of historical experiments E as a proxy for the sensitivity in future experiments. Indeed, the more dissimilar the alternatives are, the easier it is to differentiate them. Hence, we formulate the optimisation objective O1 that specifies how the interleaving parameters are optimised in Generalised Team Draft:

O1 The optimal combination of parameters $\boldsymbol{\pi}$ and $\boldsymbol{w}$ should maximise the dissimilarity D over a set of experiments E:

$$\boldsymbol{\pi}, \boldsymbol{w} = \arg\max_{\boldsymbol{\pi}', \boldsymbol{w}'} D(E, \boldsymbol{\pi}', \boldsymbol{w}')$$

This ends the framework description. In the next section, we discuss the requirement R1 in more detail.

6.5 Unbiasedness Requirement

The motivation behind R1 is to ensure that a user who clicks according to a fixed pattern that does not depend on the results shown would not provide any preference for A or B. Clearly, if R1 is not satisfied, a certain bias towards one of the alternatives might appear.

We can consider R1 from a game-theoretic perspective. Indeed, let us consider two parties: (1) a malicious user who targets to introduce a bias in the interleaving algorithm, e.g. convince us that A is better than B if it is not the case, and (2) an interleaving algorithm. The user is asked to provide a possibly randomised click sequence without knowing what result pages are going to be shown them as a result for their query. At the same time, the algorithm is asked to select an interleaving policy without knowing what click sequence the user had selected. The algorithm's goal in this game is to select an interleaving policy such that under the user's clicking sequence no preference is inferred in expectation. Indeed, the user selects how to click without knowing the results, so there cannot be any real preference. R1 formalises the criterion the policy must meet so that the algorithm achieves its goal.

To illustrate how such a bias might arise, let us consider the following "toy" example. Let us assume that the feature representation vector $\boldsymbol{\phi}(c)$ is a two-dimensional vector, with its first component $\phi_1(c)$ being equal to 1 if the clicked result is from A, and zero otherwise. Similarly, $\phi_2(c)$ is equal to 1 if a click c is performed on a result from B. Suppose we fix the interleaving policy to be uniform, and learn the vector of weights $\boldsymbol{w}$ based on the dataset of experiments. It is possible that, as a result of the learning, the weights of the features will obtain different values, e.g. if the learning dataset has more

experiments with A winning. This results in poor generalisation capabilities and biased interleaving. By considering a user who always clicks on the first position, we notice that in our toy example R1 requires w_1 to be equal to w_2.

We simplify R1 by using a restricted family of features. Namely, we use click features that do not depend on the result page[37] L_i. By restricting the set of possible features, we achieve an intuitive *antisymmetry* property: after swapping A and B ("renaming" A to B, and B to A), the experiment outcome $\Delta(e)$ will only change its sign, but not its absolute value (which is violated in our toy example). Furthermore, the following Lemma 1 shows the conditions that are sufficient to satisfy R1 if we restrict the used features:

Lemma 1 *For a feature representation $\boldsymbol{\phi}$, and a policy $\boldsymbol{\pi}$ to satisfy R1, it is sufficient that:*

- *$\boldsymbol{\phi}$ is independent from L_i;*
- *For each position p on the result page, the probability of observing a result from A must be equal to the probability of observing a result from B: $\forall p \ \sum_i \pi_i \cdot T_i(p) = 0$.*

Proof. First, using the independence of $\boldsymbol{\phi}$ from L_i, we re-write R1 as follows:

$$\sum_j \boldsymbol{w}^T \phi_j \cdot \sum_i \pi_i \cdot T_i(p_j) = 0 \tag{6.12}$$

A straightforward way to satisfy Equation (6.12) is to select $\boldsymbol{\pi}$ such that the expectation of T_i is zero for every position p:

$$\forall p \ \sum_i \pi_i \cdot T_i(p) = 0 \tag{6.13}$$

■

Lemma 1 provides us with a convenient approach to satisfy R1 while optimising the interleaving parameters. Indeed, once we use only the features that are independent from the particular interleaved result pages shown, whether R1 is satisfied or not depends only on the interleaving policy. In that case, R1 reduces to the following equality constraint:

$$R\boldsymbol{\pi} = 0 \tag{6.14}$$

where $R \in \mathbb{R}^{m \times l}$ is a matrix with its element R_{ji} equal to the team $T_i(j)$ (1 and -1 for B and A, correspondingly) of the result shown on the jth position of the interleaved result page L_i.

[37] The features cannot depend on the clicked result, its team, and its position in the original result lists of A and B. In contrast, the features can depend on the properties of the clicks itself (e.g. the position of the click, its dwell time) and the total number of clicks.

Equation (6.14) gives an intuition how the optimisation of the interleaving policy can be performed: the number of independent[38] equality constraints grows linearly as $m/2$ with the number of positions m, but the number of different team combinations $\mathbb{T}$ and thus the dimensionality of the policy vector $\boldsymbol{\pi}$ grows exponentially as $2^{m/2}$. As a result, some "degrees of freedom" appear that can be used to find a sensitive yet unbiased policy. This intuition is similar to the one behind the optimisation in Optimised Interleaving (Radlinski & Craswell, 2013).

6.6 Stratified Scoring

In Section 6.4, the experiment outcome is calculated as a sample mean of the scores of the individual interactions $\Delta(e)$, Equation (6.11). This approach is similar to the one used previously in the literature (Chapelle *et al.*, 2012; Joachims, 2003; Radlinski & Craswell, 2013; Radlinski *et al.*, 2008). Instead, we propose to use a stratified estimate $\Delta_s(e)$, where the stratification is performed according to the combination of the teams on the result pages shown to the users (possible team combinations are enumerated as $abababab$, ..., $bababababa$). As in Section 6.3.1, by $\mathbb{A}_i$ we denote the set of the user interactions where the combination of the teams on the result page shown is T_i. Using this notation, our proposed stratified estimate can be estimated as follows:

$$\Delta_s(e) = \sum_i \pi_i \cdot \frac{1}{|\mathbb{A}_i|} \sum_{a \in \mathbb{A}_i} S(a; w) \tag{6.15}$$

Both the stratified estimate $\Delta_s(e)$ and the sample mean $\Delta(e)$ have the same expected values, but the variance of $\Delta_s(e)$ can be lower and, consequently, it has higher sensitivity. Indeed, denoting the number of interactions in the experiment e as N, the variance and the expectation of the interaction score S among the sessions in the ith stratum as $var_i[S]$ and $\mathbb{E}_i[S]$, and applying the law of total variance, we obtain:

$$\begin{aligned} var\left[\Delta(e)\right] &= \frac{\sum_i \pi_i \cdot var_i[S] + \sum_i \pi_i (\mathbb{E}_i[S] - \sum_i \pi_i \cdot \mathbb{E}_i[S])^2}{N} \\ &\geq \frac{1}{N} \sum_i \pi_i \cdot var_i[S] = var\left[\Delta_s(e)\right] \end{aligned} \tag{6.16}$$

Since the frequency of T_i is determined by π_i, the probability of each stratum is known and fixed before starting an interleaving experiment.

As can be seen from Equation (6.16), the stratification reduces the variance only when the inner-strata means $\mathbb{E}_i[S]$ are different from the overall mean $\sum_i \pi_i \cdot \mathbb{E}_i[S]$. In our proposed approach of

[38] As discussed in F1 (Section 6.4, page 100), our framework relies on the Team Draft mixing algorithm. Due to its specifics, if Equation (6.13) holds for a position $2k$ and a policy π, it also holds for the position $2k+1$ and π ($k = 0, 1, ...$).

Equation (6.15), the stratification is performed according to the teams of the results on a result page T_i. In the case of the document search, T_i is a strong indicator of the outcome of a single comparison, as it specifies, for instance, if the click on the first result is counted in favour of A or B.

The stratification alone can improve the sensitivity of the interleaving experiments in some cases (see Section 6.11). Moreover, as we will discuss in Section 6.7, the use of the stratified outcome Δ_s considerably simplifies the optimisation of the interleaving parameters.

Finally, we notice that it is incorrect to ignore interactions without clicks in combination with the stratified estimator (6.15), while such an approach is sometimes used in combination with the non-stratified interleaving outcome (Equation (6.11)) and the binomial sign test, e.g. (Chapelle *et al.*, 2012). Indeed, if interactions with the top-ranked result from B are rarely clicked, the experiment result should favour A. However, the stratified estimator (6.15) would compensate this, thus adding a bias to the evaluation results. Hence, to make the evaluation set-ups uniform, we do not ignore interactions with clicks in evaluation set-ups, where Generalised Team Draft is studied (in this chapter and in Chapter 8).

6.7 Optimisation of the Parameters

To specify an instantiation of our proposed interleaving framework, we need to specify the interleaving policy $\boldsymbol{\pi}$, the feature representation $\boldsymbol{\phi}(c)$, and the vector of weights $\boldsymbol{w}$. The feature representation can be domain-specific, e.g. image search result page typically have more results, so more features can be used to encode the positions. However, our proposed approach to determine the vector of weights $\boldsymbol{w}$ and the interleaving policy $\boldsymbol{\pi}$ are the same irrespective of the domain. We adopt a data-centric approach (Yue *et al.*, 2010) to select $\boldsymbol{\pi}$ and $\boldsymbol{w}$ and select them to maximise the sensitivity on the previously collected data.

We assume that a dataset E of interleaving experiments is available, so that for each experiment in this dataset the user interactions are recorded, and the experiment outcome is known. Such a dataset can be obtained from running interleaving experiments by a search engine (e.g., Team Draft-based experiments) and selecting the experiments with a high confidence in the outcome (Chapelle *et al.*, 2012; Yue *et al.*, 2010) or by deploying "data collection" experiments where B is obtained by manually degrading A, and all possible combinations of the result lists and the team combinations are shown to the users with the uniform policy. We discuss these two approaches in more detail in Section 6.9.

To simplify the notation, without any loss in generality, we further assume that in all experiments $e \in E$ the alternative B outperformed A so that $\Delta_s(e)$ is positive. If it is not the case in a particular experiment, A and B can be swapped for that experiment.

As stated in the sensitivity optimisation objective $O1$, we want to find the values of parameters $\boldsymbol{\pi}$ and $\boldsymbol{w}$ that maximise the dissimilarity between A and B over the available experiments and satisfy constraints $R1$ and $R2$. Since the sensitivity of the interleaving does not depend on the scaling of $\boldsymbol{w}$, to make the optimisation problem well-posed, we additionally constrain $\boldsymbol{w}$ to have the unit norm. Overall, this results in a general optimisation problem of the following form:

$$\boldsymbol{\pi}, \boldsymbol{w} = \arg\max_{\boldsymbol{\pi}', \boldsymbol{w}'} D(E, \boldsymbol{\pi}', \boldsymbol{w}') \;\; s.t.\; R1, R2, \; \|\boldsymbol{w}'\|_2 = 1$$

Further, we discuss two ways to specify the idea of dissimilarity, proposed in (Yue *et al.*, 2010): the *mean score* and the *z-score* dissimilarities.

Mean score We start with the simplest case, when dissimilarity is calculated as the mean value of the stratified score:

$$D_m(E, \boldsymbol{\pi}, S) = \frac{1}{|E|} \sum_{e \in E} \sum_i \pi_i \frac{1}{|\mathbb{A}_{e,i}|} \sum_{c \in a, a \in \mathbb{A}_{e,i}} T(c) \cdot \boldsymbol{w}^T \boldsymbol{\phi}(c) \tag{6.17}$$

where $\mathbb{A}_{e,i}$ is the set of user interactions with the team combination T_i demonstrated.

Further, we introduce a matrix X with its columns corresponding to the individual features, and rows corresponding to the strata, so that the element X_{kr} is equal to the mean value of the rth feature ϕ_r in the kth stratum:

$$X_{kr} = \frac{1}{|E|} \sum_{e \in E} \frac{1}{|\mathbb{A}_{e,k}|} \sum_{c \in a, a \in \mathbb{A}_{e,k}} T(c) \cdot \phi_r(c)$$

Using the introduced notation, the optimisation objective can be re-written as follows:

$$D_m(E, \boldsymbol{\pi}, \boldsymbol{w}) = \boldsymbol{\pi}^T X \boldsymbol{w}$$

Thus, we are looking for $\boldsymbol{\pi}, \boldsymbol{w}$ that maximise the following Equation:

$$\begin{array}{l} \boldsymbol{\pi}', \boldsymbol{w}' = \arg\max_{\boldsymbol{\pi}, \boldsymbol{w}} \left[\boldsymbol{\pi}^T X \boldsymbol{w}\right] \\ s.t.\; R1, R2, \; \|\boldsymbol{w}\|_2 = 1 \end{array} \tag{6.18}$$

Finally, we notice that if we set $\boldsymbol{\pi}$ to be the uniform policy, the solution of the optimisation problem (6.18) becomes similar to the solution of the corresponding case in (Yue *et al.*, 2010): $\boldsymbol{w}$ lies on the unit sphere $\|\boldsymbol{w}\|_2 = 1$ and maximises the dot product $\boldsymbol{\pi}^T X \cdot \boldsymbol{w}$, so $\boldsymbol{w} = \frac{\boldsymbol{\pi}^T X}{\|\boldsymbol{\pi}^T X\|_2}$. The difference is in the way X is calculated, as the scores are stratified in our case.

Z-score The second way to specify the level of dissimilarity between A and B proposed in (Yue *et al.*, 2010) is to measure the z-score statistic. Informally, this measures how the distance between A and B is far from zero in terms of the variance of this distance. In contrast to Yue *et al.* (2010), we do not combine all experiments in a single artificial experiment, but optimise the mean z-score over

the set of experiments. To equalise the contribution of each experiment in the optimisation objective, we normalise the z-scores by dividing them by $\sqrt{N}$, where N is the number of interactions in an experiment. This normalised z-score indicates the rate of convergence of the experiment's outcome, i.e. it is a characteristic that reflects the interleaving sensitivity.

For a single experiment e, the normalised z-score can be calculated as follows:

$$D_z(e, \boldsymbol{\pi}, \boldsymbol{w}) = \frac{1}{\sqrt{N}} \frac{\Delta_s(e)}{\sqrt{var\left[\Delta_s(e)\right]}} \tag{6.19}$$

As in the case of the mean score dissimilarity, we introduce a matrix X with its elements equal to the per-stratum means of the individual features:

$$X_{kr} = \frac{1}{|\mathbb{A}_{e,k}|} \sum_{c \in a, a \in \mathbb{A}_{e,k}} T(c) \cdot \phi_r(c)$$

Again, the score $\Delta_s(e)$ can be found as $\boldsymbol{\pi}^T X \boldsymbol{w}$. Due to the stratified representation of the score, the variance of $\Delta_s(e)$ breaks down to a weighted sum of the per-stratum variances[39]:

$$var\left[\Delta_s(e)\right] = \frac{1}{N} \sum_i \pi_i \cdot var_i\left[S\right] = \frac{1}{N} \sum_i \pi_i \cdot \boldsymbol{w}^T Z_i \boldsymbol{w}$$

where N is the number of interactions in e, and Z_i is the covariance matrix of the interaction scores $\sum_{c \in a} T(c) \cdot \boldsymbol{\phi}(c)$ for the ith stratum:

$$Z_i = \sum_{a \in \mathbb{A}_{e,i}} \frac{1}{|\mathbb{A}_{e,i}|} \left(\sum_{c \in a} T(c) \boldsymbol{\phi}(c) - \boldsymbol{\phi}_i \right) \left(\sum_{c \in a} T(c) \boldsymbol{\phi}(c) - \boldsymbol{\phi}_i \right)^T \tag{6.20}$$

and $\boldsymbol{\phi}_i$ is the mean feature vector for the ith stratum:

$$\boldsymbol{\phi}_i = \frac{1}{|\mathbb{A}_{e,i}|} \sum_{c \in a, a \in \mathbb{A}_{e,i}} T(c) \cdot \boldsymbol{\phi}(c)$$

Overall, we obtain the following optimisation problem:

$$\begin{aligned} &\boldsymbol{\pi}', \boldsymbol{w}' = \arg\max_{\boldsymbol{\pi}, \boldsymbol{w}} \textstyle\sum_{e \in E} \frac{\boldsymbol{\pi}^T X^e \boldsymbol{w}}{\sqrt{\boldsymbol{w}^T \left(\sum_i \pi_i \cdot Z_i^e \right) \boldsymbol{w}}} \\ &s.t.\ R1, R2, \left\| \boldsymbol{w} \right\|_2 = 1 \end{aligned} \tag{6.21}$$

where X^e and Z_i^e are per-stratum means of the features and covariance matrix calculated for the experiment e.

The use of stratification considerably simplifies the form of the optimisation problem (6.21). Indeed, to calculate the variance of $\Delta_s(e)$ in the denominator of Equation (6.19) we used the right part of the inequality (6.16). In the non-stratified case, the variance is represented by the left part of (6.16). The

[39] We assume that they have converged to their true values.

Table 6.2: Datasets statistics.

Domain	# exp.	A $\succ$ B	mean # interactions	median # interactions
Document	145	84	450K	279K
Image	5	5	38K	34K

latter case is harder for the optimisation due to additional mutual dependencies of the variables (e.g. the variance becomes a third-order polynomial w.r.t. $\boldsymbol{\pi}$, while it is linear in the stratified case).

In contrast to the case considered by Yue et al., there is no closed-form solution to the problems (6.18) and (6.21) (due to the additional variable $\boldsymbol{\pi}$, the requirements R1 and R2, and the summation over all experiments). Instead, we optimise (6.18) and (6.21) numerically, using the Sequential Least SQuares Programming (SLSQP) routine implemented in scipy[40] (Jones *et al.*, 2016). As an initial approximation, we use the uniform policy and the solution of the corresponding problem in (Yue *et al.*, 2010).

6.8 Datasets

In our evaluation study we use two datasets: a dataset of Team Draft-based document search online experiments performed by Yandex, and a dataset of preliminary interleaving experiments performed on the image search service of the same company. We discuss them in more detail below.

Document search We build the dataset of the Team Draft-based online experiments as follows. First, we randomly sample a set of 336 interleaving experiments performed by Yandex in the period from January to November, 2014. These experiments test changes in the search ranking algorithm that were developed as a part of the search engine's evolution. The experiments also differ by country, and the geographical region they are deployed on. From these experiments we selected 145 experiments where the winner (A or B) is determined with a high level of confidence, $p \leq 0.01$ (binomial sign test, deduped click weighting scheme (Chapelle *et al.*, 2012)).

Overall, we believe that our dataset of document search interleaving experiments is one of the largest used in the literature. For instance, the dataset used by Schuth *et al.* (2015) contained 38 interleaving experiments, and 23 interleaving experiments were used in (Chapelle *et al.*, 2012). We will use the same dataset of interleaving experiments in Chapter 7 to assess the sequential testing approaches.

Image search In contrast to the web document search case, a representative set of online interleaving experiments is not available to us. Instead, we take five "data collection" experiments. In each of these

[40] `http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.optimize.fmin_slsqp.html`

experiments, the evaluated ranker B is obtained by degrading A in a controlled manner. After that, the corresponding "comparison" of A and B is deployed. In these "experiments" the interleaved result pages are obtained by interleaving the ranked lists returned by A and B, as discussed in Section 6.4, and showing them with the uniform policy (i.e. applying Team Draft). The following modifications of the production ranker to generate the alternative system B were used:

- swapping the results ranked as 1..15 with the results ranked 16..30;
- random permutation of the top-ranked results;
- promoting results with a low resolution;
- setting an important subset of the ranking features to zero;
- randomly ignoring some subsets of the search index.

As a result, we obtained a dataset of image search experiments, which can be used to adjust the interleaving parameters $\boldsymbol{w}$ and $\boldsymbol{\pi}$, as discussed in Section 6.7. While the modifications used are severe and might make the difference between the tested systems easy to detect, this is the only dataset available for us. Further, as our work in this chapter is the first to study the application of interleaving in the image search domain, a more representative dataset was never discussed in the literature.

We provide descriptive statistics of both datasets in Table 6.2.

6.9 Instantiation

As discussed in Section 6.7, what changes for different domains is the feature representation of the clicks ($\boldsymbol{\phi}(c)$). Further we describe what features we use in our experimental study. All features we use are independent from the result page presented, so they meet the requirements of Lemma 1.

Document search features For each click in a user interaction, we calculate a set of 24 features, split into four families: Rank-based, Dwell time-based, Order-based, and Linear score-based features. We report these features along with their descriptions in Table 6.3. Generally, these features are similar to those used by Yue *et al.* (2010).

Image search features The click features we use for image search interleaving are similar to the features used for document search. We exclude some rank-based features, as they are not meaningful for the two-dimensional result presentation (e.g. feature #11 assumes that the users tend to examine results in a rank-wise order). The full list of features used for the image search click representation is provided in Table 6.4.

Table 6.3: Click features for document search.

Feature family	id	Description
Rank-based		Transformations of the click's rank, normalised by the number of clicks
	1-10	position indicators, $f_i = \mathbb{I}\{rank = i\}$
	11	rank
	12	$\sqrt{rank}$
	13	$log(rank)$
	14	$\mathbb{I}\{rank > 4\}$
	15	$\mathbb{I}\{rank > d\}$, where d is the number of identical results in the tops of A and B
Dwell time-based		Indicators of the dwell time (seconds), normalised by the number of clicks
	16	$\mathbb{I}\{dwell \leq 30\}$
	17	$\mathbb{I}\{dwell \in (30, 60]\}$
	18	$\mathbb{I}\{dwell \in (60, 90]\}$
	19	$\mathbb{I}\{dwell \in (90, 120]\}$
	20	$\mathbb{I}\{dwell > 120\}$
Order-based		Indicators of the click's position in the interaction
	21	is the click first
	22	is the click last
Linear score-based		after applying the scoring rule F4, these features represent the (normalised) number of clicks the results from B received
	23	$f_{23} = 1$
	24	$f_{24} = 1/n$, where n is the total number of clicks

Stratification In the document search scenario, we stratify the estimate of the experiment outcome according to the teams of the results on the first result page. This gives us $2^{10/2} = 32$ strata. The same strata are used for the policy optimisation: the policy specifies the probability of using a specific team combination to generate the first interleaved result page. The remaining pages are generated using the standard Team Draft procedure, and it can be shown that the interleaving is unbiased in terms of R1 in that case.

In the case of image search, the stratification is less straightforward. Indeed, the stratification according to the teams of the top 30 results on the first result page, will yield $2^{30/2} = 32768$ strata. On one hand, according to Equation (6.16), using more fine-grained strata results in equal or lower variance. On the other hand, to run the optimisation discussed in Section 6.7, we need to estimate the per-stratum means and covariances of the features. This results in a trade-off between an increased sensitivity due to more fine-grained stratification and a higher error of the optimisation with unreliable parameters. Thus we performed the search for the optimal number of top results to be used in stratification as a part of the training process, as discussed in Section 6.10.3.

6.10 Experimental Methodology

In our evaluation study, we aim to answer the following research questions:

RQ6.1 Is our Generalised Team Draft framework more sensitive than the baselines on the document and image search data?

RQ6.2 What aspects of the sensitivity optimisation (stratification, credit assignment and policy optimisation) contribute most to the increased sensitivity?

To answer these questions, we firstly describe the baselines we use in Section 6.10.1. After that, we introduce the metric we use in Section 6.10.2. Finally, we describe the evaluation methodology in Section 6.10.3.

6.10.1 Baselines

In our study, we compare the sensitivity of our proposed Generalised Team Draft framework (Section 6.4) to the Team Draft algorithm with the credit assignment functions varied. We consider credit assignment functions of two types: the heuristic click weighting schemes that are applicable for Team Draft and considered in (Chapelle *et al.*, 2012), and the learned scoring functions trained according to the approach in (Yue *et al.*, 2010). All these baselines are non-stratified.

Linear In the simplest scoring scheme, we calculate the difference in the number of clicks on the results from A and B:

$$S(a; \boldsymbol{w}) = \sum_{c \in a} T(c)$$

Normalised Linear In the *Normalised Linear* scheme, the score of B in a particular interaction is normalised by the number of clicks in this interaction:

$$S(a; \boldsymbol{w}) = \frac{1}{|a|} \sum_{c \in a} T(c)$$

Binary Another approach to aggregate clicks in a single impression is to assign a unit credit to the alternative that received more clicks:

$$S(a; \boldsymbol{w}) = sign \left(\sum_{c \in a} T(c) \right)$$

Deduped Binary In the web document search scenario, it is often assumed that the users examine result lists from top to bottom. In that case, if the top k documents are identical both in A and B, all the interleaved lists have the same top k results, too. Thus, clicks on these top k results add a zero mean additive noise to the difference between the number of clicks A and B receive. A useful trick is to ignore such clicks. We combine this approach with the binary aggregation scheme:

$$S(a; \boldsymbol{w}) = sign \left(\sum_{c \in a} T_d(c) \right)$$

where $T_d(\cdot)$ is a modified team indicator function, equal to zero if the click is performed on one of the top results, identical for A and B, and equal to $T(\cdot)$ otherwise. The deduped binary scheme is one of the most sensitive schemes in the literature (Chapelle *et al.*, 2012).

Learned-mean, Learned-z In contrast to the above discussed credit assignment functions that are based on intuitive considerations, *Learned-mean* and *Learned-z* are machine-learned credit assignment functions that are based on the approach in (Yue *et al.*, 2010). These baselines use the same feature representations as our proposed interleaving framework. However, the optimisation of the interleaving policy is not performed, and it is fixed to be constant and uniform (as in Team Draft). *Learned-mean* selects the vector of weights $\boldsymbol{w}$ such that the differences between A and B are maximised, and *Learned-z* maximises the z-score objective. These objectives are close to the objectives we use in Section 6.7, but they assume a non-stratified experiment outcome and the uniform policy.

It would be interesting to compare Generalised Team Draft to the Optimised Interleaving framework (Radlinski & Craswell, 2013). However, Optimised Interleaving relies on considerably larger sets of interleaved result pages, thus the datasets of Team Draft-based interleaving experiments cannot be

Table 6.4: Click features for image search.

Feature family	id	Description
Rank-based		Transformations of the click's rank, normalised by the number of clicks
	1-30	position indicators, $f_i = \mathbb{I}\{rank = i\}$
	31	$\mathbb{I}\{rank > d\}$, where d is the number of identical results in the tops of A and B
Dwell time-based		Indicators of the dwell time (seconds), normalised by the number of clicks
	32	$\mathbb{I}\{dwell \leq 30\}$
	33	$\mathbb{I}\{dwell \in (30, 60]\}$
	34	$\mathbb{I}\{dwell \in (60, 90]\}$
	35	$\mathbb{I}\{dwell \in (90, 120]\}$
	36	$\mathbb{I}\{dwell > 120\}$
Order-based		Indicators of the click's position in the interaction
	37	is the click first
	38	is the click last
Linear score-based		after applying the scoring rule F4, these features represent the (normalised) number of clicks the results from B received
	39	$f_{48} = 1$
	40	$f_{49} = 1/n$, where n is the total number of clicks

re-used to evaluate its performance. An alternative approach is to leverage the natural variation of the search engine's rankings as a source of the result pages, as used in (Radlinski & Craswell, 2013). However, in this case, the evaluation is performed on a query level, and it is restricted to be based on the head queries only. Overall, this might lead to a less representative study.

6.10.2 Evaluation Metric

In this chapter, we use the *z-score* metric that is used to measure the interleaving sensitivity on the historical data (Chapelle *et al.*, 2012). *z-score* indicates the confidence of the evaluated method in the experiment outcome, thus it serves as a proxy to measure the sensitivity of the method: a higher confidence indicates a higher sensitivity.

Assuming that $\Delta_s(e)$ is normally distributed[41] and using the notation introduced above, we define

[41]This assumption holds when $\pi_i \cdot N$ is large enough for all i with $\pi_i > 0$, as $\Delta_s(e)$ is a weighted sum of approximately normally distributed per-stratum sample means, thus it is normally distributed.

the *z-score* statistic on the data of the experiment e as follows:

$$Z = \frac{\Delta_s(e)}{\sqrt{var[\Delta_s(e)]}} = \frac{\Delta_s(e)}{\sqrt{\sum_i \pi_i \cdot var_i[S]}}\sqrt{N} \tag{6.22}$$

To calculate the z-score statistic for an interleaving method with a non-uniform policy on data obtained from an experiment with the uniform policy, we use the per-stratum sample estimates of the expectation $\mathbb{E}_i[S]$ and the variance $var_i[S]$ (Equation (6.16)), calculated on the experimental data, and the policy specified by the interleaving method.

The value of (6.22) indicates how far the score $\Delta_s(e)$ deviates from zero, measured by its standard deviation. Thus it indicates the confidence level of the experiment outcome and can be mapped into a p-value (under the null hypothesis the true value of $\Delta_s(e)$ is 0). For instance, Z of 1.96 (2.58) corresponds to the two-sided p-value of 0.05 (0.01).

In the case of the non-stratified estimate $\Delta(e)$, z-score is calculated similarly:

$$Z = \frac{\Delta(e)}{\sqrt{var[\Delta(e)]}} = \frac{\Delta(e)}{\sqrt{var[S]}}\sqrt{N} \tag{6.23}$$

For each interleaving experiment, we calculated the relative z-score by dividing the outcome's z-score by the z-score of the Team Draft method with the linear click weighting scheme. The relative z-score z_e has an intuitive interpretation (Chapelle *et al.*, 2012): the corresponding interleaving method needs z_e^2 less interactions in the same experiment e than the Team Draft algorithm with the linear weighting scheme to achieve the same level of confidence.

6.10.3 Experimental Methodology

In our evaluation on the document search dataset, we use 25-fold cross-validation[42]: in each split, $\frac{24}{25}$ of the interleaving experiments are used for optimisation, and the rest are used to evaluate the resulting sensitivity. The same splits are used for all the approaches that run optimisation (our proposed framework with two types of dissimilarity, and the *Learned-mean* and *Learned-z* baselines). In each split, we measure the relative z-scores of an interleaving method on the experiments in the test set. For each interleaving method, we report the overall mean and the median relative z-scores collected across all folds. We use the paired t-test on the absolute values of the non-normalised z-scores when testing the statistical significance of the performance differences.

In the case of image search, due to the smaller dataset, we replace the 25-fold cross-valuation with the leave-one-out procedure: one experiment is used for evaluation, while the others are used for training. Further, within a training step, we additionally run a nested 2-fold cross-validation procedure on

[42]We use a high number of splits, due to the high number of optimised parameters (e.g. in the document search there are 32 (policy $\boldsymbol{\pi}$) + 24 (weights $\boldsymbol{w}$) = 56 parameters and these parameters are optimised on a relatively small dataset (145 experiments). At the same time, the dataset we use is larger than any dataset of interleaving experiments used in the existing literature.

Table 6.5: Relative z-scores the interleaving outcomes for document search. The scores of the interleaving method with the highest sensitivity ($p < 0.001$, Wilcoxon test) are denoted by $\triangle$.

	Non-stratified						**Stratified**							
	Linear	NLinear	Binary	Deduped	L_m	L_z	Linear	NLinear	Binary	Deduped	L_m^s	L_z^s	F_m	F_z
Mean	1.00	1.03	1.10	2.33	1.35	2.36	1.05	1.11	1.18	2.33	1.38	2.36	1.38	**2.52**$\triangle$
Median	1.00	0.92	0.97	1.98	1.22	2.09	1.04	0.99	1.04	1.98	1.23	2.09	1.23	**2.15**$\triangle$

the training set to find the optimal number of the result teams to be considered in stratification: we progressively increased k and evaluated the performance of our proposed method when teams of the top $2k$ are used for the stratification. The search is stopped when the performance degrades. In most folds the optimal k is found to be equal to 3 (i.e., the top 6 results are used for the stratification).

6.11 Results and Discussion

In this section, we use the following notation. Linear, Normalised Linear, Binary, and Deduped Binary weighting schemes correspond to *Linear, NLinear, Binary*, and *Deduped*, respectively. L_m and L_z indicate the Learned-mean and Learned-z baselines, respectively. The instantiations of our proposed framework are referred to as F_m and F_z, when the optimisation is performed to maximise the mean difference (6.18) and the z-score (6.21) objectives, respectively.

As we are interested in evaluating the effects of the stratification and the effects of the joint optimisation individually, we additionally measure the performance of the baselines when the stratified outcome $\Delta_s(e)$ is calculated. The stratified modifications of the interleaving methods L_m and L_z are denoted as L_m^s and L_z^s. L_m^s and L_z^s use the stratified objectives we proposed in Section 6.7, and are close to our Generalised Team Draft framework with the interleaving policy fixed to be uniform.

In our experiments on both document and image search datasets, all of the studied interleaving methods correctly determined the preference for A or B, hence we conclude that the preferences obtained from Generalised Team Draft are in agreement with the preferences inferred from Team Draft.

6.11.1 Document Search

In Table 6.5 we report the results of the evaluation procedure discussed in Section 6.10.3 applied for the web document search data. In the left part of Table 6.5 (*Non-stratified* group), we report the mean and median relative z-scores for the baselines with no stratification applied. In the right part (*Stratified* group), we report the performance of our proposed framework as well as that of the baselines with the stratification applied.

On analysing the results of the non-stratified baselines, reported in the left part of Table 6.5, we notice that their relative performance is generally in line with the results reported in (Chapelle *et al.*, 2012). Indeed, the deduped binary scheme with its median relative z-score of 1.98 considerably outperforms other considered heuristic schemes: Linear (1.0), Normalised Linear (0.92), and Binary (0.97); similarly, L_z (2.09) outperforms L_m (1.22). Overall, the baseline scoring schemes can be ordered according to the mean relative z-scores as follows: *Linear* $\prec$ *NLinear* $\prec$ *Binary* $\prec$ *Deduped* $\prec L_z$.

On comparing the relative z-scores of Linear, NLinear, Binary, and Deduped with and without the stratification applied (left vs. right parts of Table 6.5), we observe that in some cases the stratification greatly increases the interleaving sensitivity. For instance, the mean and the median relative z-scores of Binary grow from 1.10 and 0.97 to 1.18 and 1.04, respectively. Noticeable improvements are obtained for all of the heuristic baseline schemes, except for Deduped, where the improvement is small. Interestingly, an improvement is also observed for L_m: its stratified modification L_m^s exhibits a mean relative z-score of 1.38, while L_m has a mean z-score of 1.35.

In all cases, the credit assignment function that optimises the mean difference between A and B performs worse than the credit assignment functions learned to maximise the z-score. For instance, L_z^s demonstrates considerably higher median relative confidence than L_m^s (2.09 vs. 1.23).

By additionally performing the interleaving policy optimisation, F_z achieves a noticeable sensitivity gain in comparison with the stratified L_z^s (medians: F_z, 2.15 vs. L_z, 2.09; means: 2.52 vs. 2.36). F_z also achieves the highest overall sensitivity, with the median relative z-score of 2.15 and the mean z-score of 2.52. This implies that an interleaving experiment that uses the interleaving method F_z requires $2.15^2 = 4.62$ times less user interactions (in median) than the non-stratified Team Draft with the linear scoring to achieve the same level of confidence. In comparison with the best performing baseline, L_z, it requires $\left(\frac{2.15}{2.09}\right)^2 = 1.06$ times less data to achieve the same level of confidence (in median). The corresponding improvement in the mean relative z-score is $\frac{2.52}{2.36} = 1.07$.

Overall, our observations allow us to answer the stated research questions RQ6.1 and RQ6.2. Our proposed Generalised Team Draft demonstrates the highest sensitivity in the case of document search (median relative z-score, F_z 2.15 vs. L_z 2.09; mean relative z-score F_z 2.52 vs. 2.36 L_z). To answer RQ6.2, we notice that the highest sensitivity gains are achieved by the click scoring optimisation (e.g. L_z 2.09 vs. *Deduped* 1.98, median relative z-score) and the policy optimisation (F_z 2.15 vs. L_z^s 2.09, median relative z-score).

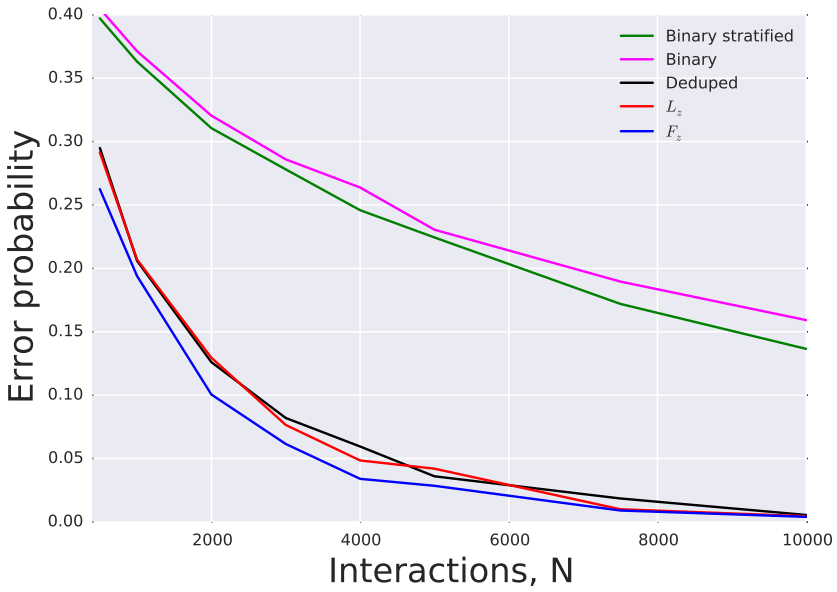


Figure 6.4: The probability that an interleaving method disagrees with the true preference, depending on the size of the sample. The document search dataset.

6.11.2 Visualisation

We illustrate the relative performance of the studied interleaving methods on the document search dataset using the following procedure. We select one experiment to be used as a test experiment, and use the remaining experiments to optimise the interleaving parameters. Further, we estimate the probability that an interleaving method disagrees with the ground truth preference in the test experiment by obtaining 5,000 samples of N user interactions. We varied N in $(500, ..., 10^4)$. For the baseline methods, N interactions are obtained by sampling from the experiment's interactions with replacement (bootstrap sampling). For F_z, the sample is obtained by firstly allocating N interactions to the strata according to a multinomial distribution specified by the policy $\boldsymbol{\pi}$, and further sampling from the individual strata (with replacement). The outcome is calculated using the stratified estimate Δ_e. This sampling process simulates the case of policy $\boldsymbol{\pi}$ to be applied in a real-life scenario. A higher error probability indicates lower sensitivity and it is related to the outcome's p-value under the bootstrap test.

In Figure 6.4, we report the obtained error probabilities. From Figure 6.4 we observe that the optimisation-based methods (F_z and L_z) dramatically increase the interleaving sensitivity and outperform the *Binary* baseline. For instance, the probability error of 0.1 is achieved by F_z with 2,000 interactions, but *Binary* requires more than 10,000 interactions to achieve the same level of error. Adding

Table 6.6: Features with the highest weights, document search.

Feature id, i	15	23	8	2	22
Description	Deduped score	Unormalised score	Rank = 8?	Rank = 2?	Last click?
Weight, w_i	0.97	0.09	0.08	-0.07	0.06

stratification to the *Binary* baseline allows us to reduce the probability of error. The deduped binary baseline performs considerably better than *Binary* and achieves performance close to the one of L_z. Among the methods that use optimisation, F_z consistently demonstrates lower probability of error than L_z. For instance, when $4,000$ interactions are used, F_z has the probability of error approximately equal to 0.03, while L_z makes an error in more than 0.05 of the samples. Overall, these observations are in line with results reported in Table 6.5. However, this illustration is also important as it does not rely on the z-score statistic.

6.11.3 Analysis of the Learned Parameters

In Table 6.6 we report the five features with highest absolute values of components of the weight vector $\boldsymbol{w}$ when trained on the whole dataset. From Table 6.6 we observe that the highest weight is paid to the deduped score feature that represents the deduped binary scoring scheme, the strongest heuristic baseline according to our experiments. Considerably less weight is assigned to the linear score feature (*Linear* baseline). Interestingly, the clicks on the first position are slightly penalised due to a negative score, indicating that these clicks might be less meaningful due to the positions bias.[43]

In Table 6.7 we report the optimised interleaving policy learned by our proposed interleaving framework on the entire dataset of the document search interleaving experiments. Clearly, the resulting policy satisfies the requirement specified by Lemma 1 (the probabilities of observing *a* and *b* are equal for each position). Since the optimisation objective stated in Equation (6.21) is essentially a linear programming problem w.r.t. the interleaving policy, the optimum is in a vertex of the feasible region, so the support of optimal policy distribution (i.e. the set of team combinations that have non-zero probability of being demonstrated) would use few team combinations.[44] This consideration holds in our case, as only two team combinations have non-zero probabilities of being demonstrated.

Finally, in these combinations teams *A* and *B* are strictly alternating in positions, thus the optimal policy is somewhat intuitively reasonable.

[43]The users trust the search engine and tend to click on the top ranked results even if they do not look useful (Craswell *et al.*, 2008).

[44]This effect is also discussed in Section 6.3 and in (Kharitonov *et al.*, 2013*c*). In case of a small training dataset, it might be harmful and can be avoided by adding a quadratic penalisation term, as in Equation (6.8a*), page 93.

Table 6.7: The optimal interleaving policy, learned on the document search dataset.

Team combination	Probability
a,b,a,b,a,b,a,b,a,b	0.5
b,a,b,a,b,a,b,a,b,a	0.5

Table 6.8: Relative z-scores of the interleaving outcomes for image search. The scores of the interleaving method with the highest sensitivity ($p < 0.05$, Wilcoxon test) are denoted by $\triangle$.

	Non-stratified						**Stratified**							
	Linear	NLinear	Binary	Deduped	L_m	L_z	Linear	NLinear	Binary	Deduped	L_m^s	L_z^s	F_m	F_z
Mean	1.00	1.03	1.05	1.17	1.11	1.17	1.00	1.03	1.05	1.17	1.11	1.18	1.14	**1.21**$\triangle$
Median	1.00	0.98	1.02	1.11	1.08	1.16	1.00	0.98	1.03	1.11	1.08	1.16	1.14	**1.18**$\triangle$

6.11.4 Image Search

In Table 6.8 we report the results of the evaluation for the case of image search. Generally, we observe that the results are similar to the document search case. The machine-learned interleaving methods that optimise the z-score objective (L_z, L_z^s, and F_z) outperform both the methods with the heuristic credit assignment (*Linear*, *NLinear*, *Binary*, and *Deduped*) and the methods that optimise the mean difference, namely L_m, L_m^s and F_m.

However, in contrast to the document search experiments, the sensitivity gains due to stratification are less noticeable on the image search data. A possible explanation is that the differences of the means of the strata are smaller than in the case of the document search. Indeed, if the users tend to examine most of the results (which is easier for images than for document snippets) and click more, then the teams of the first results are not such a strong indicator of the total credit in an interaction, as in the case of document search. Interestingly, *Deduped* is sensitive in image search, too.

The overall highest performance ($p < 0.05$) is achieved by our proposed framework with the z-score-based optimisation objective (F_z, mean relative z-score 1.21, median 1.18). This value of the metric implies that our proposed Generalised Team Draft framework requires $1.18^2 = 1.39$ less data (median) than the *Linear* baseline to achieve the same level of confidence. In comparison to the best-performing baseline L_z, the corresponding decrease is $(\frac{1.18}{1.16})^2 = 1.03$ in median. However, the difference of the means is higher (L_z, 1.18 vs. F_z, 1.21). A possible explanation for the smaller improvements is that the degradations used in our image search dataset are relatively strong and easy to detect, thus it is harder to achieve a high level of improvement over the baselines.

These observations allow us to answer the stated research questions, RQ6.1 and RQ6.2, in the case of image search. Indeed, our proposed Generalised Team Draft achieves the highest sensitivity (RQ6.1).

The gains from the stratification are almost not noticeable (e.g. L_z^s 1.18 vs. L_z 1.17, mean relative z-score), while the weight optimisation (L_z 1.16 vs. $Deduped$ 1.11, median z-score) and the policy optimisation (F_z 1.21 vs. L_z^s 1.18, median z-score) contribute more (RQ6.2).

6.11.5 Summary

Our evaluation study in Section 6.11 allows us to answer the research questions we stated in Section 6.10. Our proposed framework achieves the highest sensitivity on both the document search and the image search datasets (RQ6.1). On the document search data, the non-stratified optimised interleaving method L_z (median 2.09 and mean 2.36 z-scores) outperforms the best non-learned baselines (*Deduped Binary*, median 1.98 and mean 2.33). Further, our proposed F_z additionally performs the policy optimisation and achieves the highest median relative confidence level (median of 2.15 and mean 2.52). In contrast, on the image search data the improvements are relatively smaller. However, gains in the sensitivity are still obtained by the credit assignment learning (L_z, 1.16 vs. *Deduped*, 1.11) and the policy optimisation (F_z, 1.18 vs. L_z^s, 1.16).

In many cases, stratification improves performance (e.g. the median z-scores of *Binary* grow from 0.97 to 1.04, Table 6.5). In other cases, the improvements are relatively small (L_m has its median score of 1.23 in contrast to L_m^s with median of 1.22, Table 6.5).

These observations answer RQ6.2: the credit assignment and the policy optimisation increase the interleaving sensitivity on both datasets; our proposed stratification has higher impact on the interleaving sensitivity in the document search than in the image search domain.

6.12 Conclusion

In this chapter we addressed the important problem of improving the interleaving sensitivity which has a direct effect on the evaluation efficiency: given an evaluation method with a higher sensitivity, one can deploy online experiments for a shorter period of time.

We started with discussing a user model-based approach that optimises interleaving policy so that the sensitivity is increased on the per-query level. In a small-scale qualitative study we observed that this approach can be fruitful. However, it has considerable limitations, particularly from the practical application point of view. These considerations led us to develop a superior interleaving framework, Generalised Team Draft, which also uses historical interaction data in the form of experimental datasets, but overcomes the limitations of the user model-based approach.

Generalised Team Draft generalises the research in the exiting literature in two aspects. First, it achieves an increased sensitivity by performing a joint optimisation of the credit assignment function

and the interleaving policy. Second, it is formulated to be general with respect to the way the results are presented, thus it can be applied in the domains with the grid-based representation, such as image search. Further, to simplify the optimisation procedure, we proposed to use a stratified estimate of the experiment outcome (Section 6.6). This stratification is useful on its own, as in some cases it reduces the variance of the experiment outcome and thus increases the sensitivity. Finally, we proposed a generalised unbiasedness requirement R1 (Section 6.4) that the feature-based credit assignment and the interleaving policy have to meet for the interleaving to be unbiased.

In our evaluation study, we used two datasets of the Team Draft-based experiments obtained from Yandex. The first dataset contains 145 interleaving experiments performed in the document search domain, and the second dataset contains 5 "data collection" experiments deployed for image search. In our study, we demonstrated that our proposed Generalised Team Draft framework achieves the highest sensitivity on both datasets. Specifically, we observe that Generalised Team Draft requires up to 1.06 times (median) less data than the top-performing baseline on the document search dataset (Table 6.5: F_z 2.15 vs. L_z 2.09), and up to 1.03 times (median) less data on the image search dataset (Table 6.8: F_z 1.18 vs. L_z 1.16).

Overall, our study in this chapter follows the roadmap for improving the efficiency of the evaluation pipeline we described in Section 3.5. Indeed, in this chapter we discussed how the interleaving sensitivity can be improved (a) by using user model-based interleaving policy optimisation or (b) by using Generalised Team Draft, which performs a joint optimisation of interleaving policy and the click scoring scheme. An increased interleaving sensitivity allows a search engine deploy online experiments for a shorter period of time, thus increasing the efficiency of the evaluation pipeline.

At the same time, both the click model-based approach (Section 6.3) and Generalised Team Draft (Section 6.4) optimise interleaving parameters against a dataset of historical interaction data. Indeed, in the user model-based approach we used historical interaction data to train a click model that was used in the optimisation process. Generalised Team Draft adjusts the interleaving parameters to achieve a highest sensitivity on a set of historical experimental data. Hence, we conclude that our study in this chapter supports the statement of this thesis (Section 1.3): by re-using historical interaction data, we increase the efficiency of the evaluation pipeline.

In the next chapter, we discuss how the efficiency of the interleaving experiments can be improved by using sequential statistical tests. We demonstrate that such tests can stop online experiments earlier (on average) and hence increase the efficiency of the evaluation pipeline. After that, in Chapter 8, we demonstrate that in fact the sensitivity optimisation methods discussed in this chapter can be com-

bined with sequential testing methods and this combination can result in even higher gains in evaluation efficiency.

Chapter 7

Sequential Testing for Early Stopping of Interleaving Experiments

7.1 Introduction

As discussed in Section 2.4, each online experiment, performed by means of A/B testing or interleaving, typically lasts for a considerable time period, usually about a week or two. As a result, the efficiency of the evaluation pipeline is bounded by the efficiency of the online evaluation step. This limitation was addressed earlier, primarily by optimising the sensitivity of the online evaluation methods (Chapter 6). However, the possibility to reduce the duration of the interleaving experiments by appropriate use of sequential hypothesis testing procedures has not been studied in the literature. Such testing procedures perform interim stops during an experiment and decide if the experiment should be continued or a decision can be made based on the already observed data. In this chapter, we aim to close this gap and we will show that such testing procedures can significantly reduce the mean deployment time of interleaving experiments. We demonstrate that by modifying two sequential testing procedures to make them applicable for interleaving, we obtain a considerable improvement in the average time an online experiment takes. This chapter is based on a publication (Kharitonov, Vorobyev, Macdonald, Serdyukov & Ounis, 2015).

We study a modification of the O'Brien & Fleming repeated significance test (O'Brien & Fleming, 1979), and several possible modifications of the MaxSPRT test proposed by Kulldorff *et al.* (2011). From our experimental study in Section 7.6, we will observe that the highest efficiency is achieved by a MaxSPRT-based test with its stopping threshold adjusted on a set of historical online experiments where a search system is compared to itself (A/A experiments). Consequently, in this chapter we demonstrate that by using historical interaction data we achieve an improvement in the efficiency of the

evaluation pipeline, hence our work in this chapter supports the statement of this thesis (Section 1.3) and corresponds to the third point of our roadmap for improving the evaluation pipeline, discussed in Section 3.5.

Overall, the contributions of this chapter are two-fold:

- We propose several sequential testing methods that reflect the distributions of the data generated in interleaving experiments, and describe how to adjust their stopping thresholds based on query log data;
- We perform an extensive evaluation study of the performance of our proposed methods using a real-life dataset of interleaving experiments.

The remainder of this chapter is organised as follows. In Section 7.2 we discuss the related work. In Section 7.3 we introduce methods for performing sequential statistical analysis for interleaving experiments. A dataset used in our empirical study is described in Section 7.4. Our evaluation methodology and the results we obtained are described in Sections 7.5 and 7.6, respectively. We conclude this chapter and discuss future work in Section 7.7.

7.2 Related Work

Our work in this chapter is closely related to two areas of research. The first area is concentrated on developing approaches to speed up the existing online evaluation methods, which has been discussed in details in Section 6.2. However, all these approaches share the same statistical testing scenario, discussed in Section 2.3. In this scenario, an online evaluation experiment is deployed. After running this experiment for a pre-defined period of time (e.g. a week), the experiment is stopped. Next, some form of statistical test, such as the binomial test in the case of interleaving, is performed on the collected user interaction data to infer if a statistically significant difference between the tested alternatives was observed.

Importantly, in this scenario, each experiment is deployed for a period of time that is fixed before the experiment starts. However, it is likely that some experiments will compare highly contrasting alternatives, and in such cases it should be possible to stop the experiment early and still be able to reliably detect user preferences between the compared alternatives. The sequential analysis methods allow us to do that, and they form the second related area of research.

Sequential statistical testing appeared to address the demands of the military testing during World War II, resulting in Wald's *sequential probability ratio test* (SPRT) (Wald, 1945). This test performs an

analysis in steps. At each step, a new data point is considered, and the decision is taken if the observed data is enough to make a reliable conclusion about the considered hypotheses and the experiment should be finished, or more measurements are required.

Despite its simplicity, SPRT was shown (Wald & Wolfowitz, 1948) to be optimal when comparing two simple alternatives. Further research was conducted to improve the SPRT-based methods in a variety of directions. For instance, the 2-SPRT test was proposed to minimise the expected sample size at a specified parameter when discriminating hypotheses $H_0 : \theta = \theta_0$ against $H_1 : \theta = \theta_1 > \theta_0$ (Bartroff *et al.*, 2012). Another SPRT-based test that can be used to test against a complex hypothesis, MaxSPRT (Kulldorff *et al.*, 2011), was proposed for the post-approval drug safety surveillance. These tests can be applied when the parameters of the alternative hypothesis are not known before running the experiment. A similar problem arises while running online experiments. Indeed, in interleaving experiments the null hypothesis can be specified (both evaluated algorithms are equally likely to win in a particular user session), but the difference between the evaluated algorithms is hard to estimate before running an experiment.

As an alternative to the SPRT-based tests, "repeated significance tests" (RST) were proposed. As conventional single-sample tests are often used in clinical trials, the motivation behind RST is to apply them repeatedly during the trial. These tests evolved into a group sequential RST (Pocock, 1977), where the data is accumulated between tests. Further, a group sequential testing procedure that had a better performance was proposed in (O'Brien & Fleming, 1979). These methods became popular within clinical trials, as the sequential procedures reduce the time the participants are exposed to ineffective or harmful treatments.

Thereafter, the group sequential testing approach was intensively developed. Wang & Tsiatis (1987) suggested a parametric family of tests, which generalises both Pocock and O'Brien & Fleming tests, and can be optimised to be nearly optimal w.r.t. a fixed expected difference between the alternatives. Another improvement was proposed in (Lan & DeMets, 1983), which accounts for some specifics of the clinical trials: the number of subjects available between interim stops is not known in advance and can vary greatly.

Johari *et al.* (2015) considered sequential tests in A/B testing scenario. They introduced the notion of *always valid* p-values that provide valid statistical inference whenever an experiment is stopped. Further, Johari *et al.* (2015) proposed the mixture sequential probability ratio test and demonstrated how their results can be extended to account for the case of multiple testing.

Overall, sequential testing is a highly developed discipline, and a variety of tests that differ by their properties and assumptions was proposed. Review can be found in (Bartroff *et al.*, 2012; Siegmund,

1985). Due to their properties, we select the O'Brien & Fleming and MaxSPRT tests as a foundation for our study in this chapter. These tests are very practical, as they do not require any prior assumptions about the expected effect size or its boundaries. The O'Brien & Fleming test can be interpreted as a repeated standard Pearson's chi-square test with progressive stopping thresholds, so that the last threshold is close to the one test scenario, which is very appealing from the practical perspective. Similarly, the MaxSPRT-based tests do not require any pre-experimental knowledge, and their decisions are extremely transparent.

Finally, to the best of our knowledge, the work of Kohavi *et al.* (2013) is the only one that mentions the use of sequential testing procedures in online web search evaluation. In their work, Kohavi et al. reports that the O'Brien & Fleming test is used in Bing to abort A/B experiments early when a severe degradation in metrics is observed. In contrast, we propose modifications of the MaxSPRT tests for interleaving, and modify the O'Brien & Fleming test for the interleaving evaluation. Moreover, we perform a thorough evaluation of the usefulness of the considered tests.

7.3 Sequential Testing for Interleaving

As we discussed in Chapter 6, several approaches to aggregate the user clicks into a credit obtained by A (the baseline system) and B (the tested system) were proposed (Chapelle *et al.*, 2012; Radlinski & Craswell, 2010; Yue *et al.*, 2010). In this chapter, we experiment with the binary aggregation scheme; experimental study with more advanced schemes, including the ones considered in Chapter 6, are performed in Chapter 8. Under the binary scheme, in each interaction, the alternative with the most results clicked receives a unit credit and is referred to as the winner. If in a session both A and B obtained an equal number of clicks, the session is considered as a tie. This scheme is similar to the binary deduped scheme considered in Section 2.3, but the clicks on identical top results are not ignored.

In order to make session outcomes binary, ties are broken randomly and sessions with no clicks are ignored (Chapelle *et al.*, 2012). Let us denote a variable Δ that represents the probability of B winning over A (i.e. getting more clicks) in a session with a click. After running an experiment, an estimate $\hat{\Delta}$ of Δ can be calculated:

$$\hat{\Delta} = \frac{w^B + \frac{1}{2}t}{w^A + t + w^B} \tag{7.1}$$

where w^B and w^A denote the number of interactions where B and A win, respectively; t is the number of ties.

The goal of the statistical analysis methods we discuss in Section 7.3 is to compare two statistical hypotheses, H_0 (A and B are equally likely to win a particular impression) and H_1 (the chances to win

differ):

$$H_0 : \Delta = \frac{1}{2}, \;\; H_1 : \Delta \neq \frac{1}{2} \tag{7.2}$$

In this section, we introduce the sequential testing procedures we consider in this chapter. We start by describing the procedures applicable for interleaving experiments: O'Brien & Fleming's sequential test, modified for interleaving (OBF-I), and the MaxSPRT test.

7.3.1 OBF-I Interleaving Test

Initially, the O'Brien & Fleming's sequential test described in (O'Brien & Fleming, 1979) was formulated for clinical trials that compare two treatments on two different groups of participants. In contrast, in interleaving experiments only one group of users is used. Below we describe our adaptation OBF-I of the OBF test to the case of interleaving experiments.

In general, the test operates as follows. Firstly, the number of interim stops is specified. At these stops, the accumulated experimental results are analysed, and a decision is made if the experiments can be stopped or should be continued for further analysis. The interim stops can be specified in terms of the number of events that take place between them. For instance, the analysis can be performed every 10,000 interactions and there are up to 10 interim stops.

However, the interaction number-based formulation is not convenient in a web search set-up. Indeed, it is hard to predict the exact numbers of interactions that can take place during, e.g. a week. Instead, throughout this chapter, we use time-based stops. In other words, we assume that each experiment is constrained by a maximal time period it can be deployed for and interim stops are performed according to the time spent from the previous stop. For instance, an experiment can be deployed for up to a week and interim stops are performed every 24 hours. Such a set-up is very practical and easy to operate for practitioners. However, it also causes challenges for the sequential testing, as the number of events between interim stops might vary.

Assume that the number of possible stops, where a sequential test is allowed to analyse accumulated data and make a decision, is set to N. Let us introduce a random variable x that is equal to 1 (-1) if B (A) wins the comparison in an interaction, and 0 if a tie is observed. By x_j we denote the realisation of x observed in the jth interaction. Further, we denote the number of interactions between the $(i-1)$th and ith stops as K_i. Under the null hypothesis, the probabilities of winning a comparison in an interaction for A and B are equal. Thus, according to the central limit theorem, the normalised mean R_i of the realisations x_j observed between the $(i-1)$th and ith stops approaches the standard normal distribution

as K_i grows:

$$R_i = \frac{(x_1 + .. + x_{K_i})}{(K_i \cdot D[x])^{\frac{1}{2}}} \sim \mathbb{N}(0,1) \tag{7.3}$$

where $D[x]$ is an estimate of the variance of x.

Further, we denote the total number of interactions that occurred before the ith stop as T_i ($T_i = \sum_{j<i} K_j$), and the accumulated number of comparisons won by A (B) before the ith stop as w_i^A (w_i^B). Assuming that the number of the interactions occurring between the stops is approximately the same and equal to K, we define the accumulated statistic $O_i = (\frac{1}{\sqrt{i}} \sum_1^i R_j)^2$. Since $\sum_1^i R_j$ is a sum of variables that are distributed according to $\mathbb{N}(0,1)$, their scaled sum $\frac{1}{\sqrt{i}} \sum_1^i R_j$ also has the standard normal distribution. Thus, as a square of a standard normal variable, O_i is distributed according to the chi-squared distribution with one degree of freedom:

$$O_i = \left(\frac{1}{\sqrt{i}} \sum_{j=1}^{i} R_j \right)^2 = \frac{(w_i^B - w_i^A)^2}{iK \cdot D[x]} = \frac{(w_i^B - w_i^A)^2}{T_i \cdot D[x]} \sim \chi^2(1) \tag{7.4}$$

The estimate of the variance $D[x]$ of the outcome variable x is:

$$D[x] = \frac{1}{T_i - 1} \sum_{j=1}^{T_i} (x_j - \bar{x})^2, \;\; \bar{x} = \frac{w_i^B - w_i^A}{T_i} \tag{7.5}$$

A progressive decision criterion proposed by O'Brien & Fleming (1979) is defined as follows: at the ith stop, O_i is compared to a threshold $\frac{1}{i}\hat{O}$ that decreases at each stop ($\hat{O}$ depends on the number of stops and required Type I error). This ensures an intuitive requirement that to terminate an experiment earlier, one needs to have a higher confidence in H_1.

In an equivalent, but more convenient formulation, at each stop, a statistic $i \cdot O_i$ can be considered, and compared to a single fixed threshold $\bar{O}$. Once $i \cdot O_i$ exceeds $\bar{O}$, the experiment is terminated, and H_1 is accepted. To infer the experiment outcome, the difference between w_i^A and w_i^B is used (i.e. if $w_i^B > w_i^A$ then $B \succ A$). If at the last stop $N \cdot O_N$ still does not reach $\bar{O}$ then the hypothesis H_0 is accepted.

For the cases of small numbers of stops (less or equal to 5), the values of the threshold $\bar{O}$ can be found in Table 1 in (O'Brien & Fleming, 1979). Since in our experiments we use a higher number of stops, we briefly review how the threshold can be obtained from running Monte-Carlo simulations. The general idea is to replace R_i with random numbers generated from $\mathbb{N}(0,1)$, and adjust $\bar{O}$ so that the test will detect a difference in the α (required Type I error level) fraction of the generated tests. Formally, to perform one iteration of the simulation, we sample N (N is the required number of stops) random numbers from the standard normal distribution ($U_1, ..., U_N \sim \mathbb{N}(0,1)$), and calculate the maximum

square of their partial sums $U_m^2 = \max(U_1^2, (U_1 + U_2)^2, ..., (U_1 + ... + U_N)^2)$. We collect these maximums over 10,000 simulations. Finally, we select a value that corresponds to $(1 - \alpha)$ percentile of these maximums.

A possible heuristic is to replace the estimate of the variance $D[x]$ with its upper bound 1.[45] While this substitution might increase the time required for O_i to achieve the threshold $\bar{O}$, it also makes the decision rule even simpler: at each stop i, a normalised square of the difference between the wins of A and B is multiplied by the number of the current stop and compared to the threshold $\bar{O}$. We refer to a rule with this heuristic applied as **OBF-I***.

Before proceeding to the second family of tests proposed in Section 7.3.2, we want to stress two observations. Firstly, while we formulate the OBF-I and OBF-I* tests for binary-distributed outcomes, they can be naturally adopted for testing that the mean of non-binary outcomes is zero. This can be achieved by using non-binary observations[46] x_j in Equation (7.3). Secondly, it is important that both OBF-I and OBF-I* assume that the number of sessions performed between stops is large enough so that the central limit theorem can be applied.

7.3.2 MaxSPRT-based Test

At the core of the SPRT family of tests (Kulldorff *et al.*, 2011; Wald, 1945) is the likelihood ratio statistic. Informally, this statistic equates to the likelihood of the observed data under the alternative hypothesis H_1 divided by the likelihood of the data under H_0. Once this ratio becomes big enough, H_0 can be rejected. To formalise this idea, we use the same notation as in Section 7.3.1. By T_i we denote the total number of sessions before the ith stop, w_i^A (w_i^B), and t_i are the numbers of sessions where A (B) wins, and the number of sessions with ties, respectively. Further, by m_i we denote our estimate of the number of comparisons won by B after breaking the ties:

$$m_i = w_i^B + \frac{1}{2} t_i$$

Under this notation, the logarithm of the likelihood statistic can be specified as follows:

$$L_i = \log \frac{p_1^{m_i}(1 - p_1)^{T_i - m_i}}{p_0^{m_i}(1 - p_0)^{T_i - m_i}} \tag{7.6}$$

where p_0 and p_1 are probabilities of B winning in a comparison in an interaction under H_0 and H_1, respectively. Under the null hypothesis, the alternatives are equally likely to win, so p_0 equals $\frac{1}{2}$.

[45] A unit variance is achieved if on each tie a coin is tossed and a unit credit is assigned to a random alternative. However, this might be a good approximation since for a tie to occur at least two results must be clicked, which happens rarely.

[46] We discussed some of the possible non-binary metrics in Chapter 6, e.g. Generalised Team Draft (Section 6.4) assigns clicks with real-valued scores.

Input: Type I error tolerance α, a set of A/A experiments Q.
Output: $\bar{L}$ threshold.
The vector of the ratio values observed in experiments
$Ls \leftarrow \emptyset$
Iterate over experiments
foreach $e \in Q$ **do**
  Iterate over interactions in e
  $L_m \leftarrow 0$
  foreach $i \in 1..|e|$ **do**
    $T_i \leftarrow w_i^B + t_i + w_i^A, \;\; m_i \leftarrow w_i^B + \frac{1}{2}t_i$
    Find the max. likelihood estimate $\hat{p}_1^i$ of p_1
    $\hat{p}_1^i \leftarrow \frac{1}{T_i}\left[w_i^B + \frac{1}{2}t_i\right]$
    $L_i \leftarrow \log \frac{(\hat{p}_1^i)^{m_i}(1-\hat{p}_1^i)^{T_i-m_i}}{p_0^{m_i}(1-p_0)^{T_i-m_i}}$
    Update the maximum value of L_m for the current experiment
    $L_m \leftarrow \max(L_m, L_i)$
  end
  $Ls \leftarrow Ls \bigcup \{L_m\}$
end
Retrieve the $(1-\alpha)$ percentile
$Ls \leftarrow sorted(Ls)$
$\bar{L} \leftarrow Ls[|Ls| \cdot (1-\alpha)]$

Algorithm 7.1: Learning the $\bar{L}$ threshold for MaxSPRT from a dataset of A/A experiments.

However, it is hard to specify p_1 before actually running the experiment. An intuitive idea is to replace p_1 with the maximum likelihood estimate $\hat{p}_1^i$, based on the experimental data observed before the ith stop. Informally, by estimating $\hat{p}_1^i$ we choose H_1 that is the most likely to be accepted in comparison with H_0. This idea was proposed and studied by Kulldorff *et al.* (2011) for the Poisson and Binomial distributions, and resulted in a test called MaxSPRT. Under our notation, the maximum likelihood estimate of p_1 at the ith step is:

$$\hat{p}_1^i = \frac{1}{T_i}\left(w_i^B + \frac{1}{2}t_i\right)$$

At each stop, $\hat{p}_1^i$ is estimated, and is used to substitute p_1 in L_i (Equation (7.6)). After that, L_i is compared to a pre-defined threshold $\bar{L}$. If $L_i \geq \bar{L}$, then the experiment is stopped, and H_1 is accepted. If $\hat{p}_1^i > \frac{1}{2}$ ($\hat{p}_1^i < \frac{1}{2}$) then it is inferred that $B \succ A$ ($A \succ B$). If $L_i < \bar{L}$, the experiment is continued. H_0 is accepted if the experiment reaches a pre-defined maximum length, without achieving $\bar{L}$.

To specify the threshold $\bar{L}$, the Monte-Carlo method was used in (Kulldorff *et al.*, 2011), where a series of Binomial samples were generated. However, as we will further discuss in Section 7.6, in the case of the interleaving experiments where the ties are interpreted according to Equation (7.1), this Monte-Carlo threshold adjustment is suboptimal, as it generates data with variance higher than observed in experiments.

Instead, we propose to train the threshold $\bar{L}$ on a set of experiments where a system is compared with itself. As we will discuss further in Section 7.4, such experiments are referred to as A/A experiments. Intuitively, a statistical test with Type I error set to α should detect differences in A/A experiments with the probability α. Using this idea, we adjust the threshold $\bar{L}$ so that $\bar{L}$ exceeds all values of L_i in $(1-\alpha)$ of the A/A experiments. A formal description of the optimisation of the threshold can be found in Algorithm 7.1.

Further, the MaxSPRT test where the threshold $\bar{L}$ is trained using Monte-Carlo simulations is denoted as **MaxSPRT-I-MC**. The test where the threshold $\bar{L}$ is trained using the A/A comparisons is referred to as **MaxSPRT-I-AA**.

The above proposed tests, namely MaxSPRT-I-MC and MaxSPRT-I-AA, are designed for the binary-distributed metrics, i.e. it is supposed that in each event a binary outcome is observed. However, as we have seen in Chapter 6, some of the click aggregation schemes are not binary (e.g. the machine-learned scoring schemes that assign real-valued weights to the click features, proposed in Section 6.4). Thus, we additionally introduce the **MaxSPRT-I-AA-N** test that can be used in conjunction with non-binary metrics.

The design of MaxSPRT-I-AA-N combines ideas from the MaxSPRT-I-AA and OBF-I tests. Firstly, with each outcome we associate a score, x. In general, x can be a real-valued variable, but in case of the binary credit assignment scheme we consider it to take values in $\{-1, 0, 1\}$, corresponding to A winning, a tie, or B winning in an interaction, respectively. This encoding is the same as in the OBF-I test (Equation (7.3)).

Assuming that the central limit theorem can be applied[47], the sample mean $\bar{x}$ is normally distributed. Under the null hypothesis, it is distributed as a normal variable with zero mean and some variance σ^2. Under the alternative hypothesis, it is distributed with some mean μ and variance σ^2.

The SPRT statistics on the ith stop is the log-likelihood ratio of these two hypotheses, just as in Equation (7.6):

$$L_i = \log \frac{\phi(\bar{x}^i; \mu_i; \sigma_i^2)}{\phi(\bar{x}^i; 0; \sigma_i^2)} \tag{7.7}$$

where $\bar{x}^i$ denotes the mean of the metric before the ith stop and $\phi(x; \mu; \sigma^2)$ is the probability density function of the normal distribution $\mathbb{N}(\mu, \sigma^2)$. Since both the mean μ and σ^2 are not known, we use their maximum likelihood estimates at the ith stop instead:

$$\hat{\mu}_i = \bar{x}^i; \quad \hat{\sigma}_i^2 = \frac{1}{T_i} D[x]; \quad D[x] = \frac{1}{T_i - 1} \sum_{j=1}^{T_i} \left(x_j - \bar{x}^i\right)^2 \tag{7.8}$$

[47] In particular, it is required that a sufficiently large sample of observations is collected before applying the test.

Again, T_i denotes the total number of observations occurred before the ith stop. The stopping thresholds on the L_i statistics are trained on a set of A/A experiments, similarly to the MaxSPRT-I-AA test.

The MaxSPRT tests assume that the data points arrive one-by-one, which might be impractical in modern large-scale web search engines. Indeed, an infrastructure is needed that is capable of providing a near real-time stream of individual comparisons. It might be easier to implement the data delivery in batches, e.g. each batch of data corresponding to an hour or a day of the user activity. Since the discussed tests can be applied to analyse batch data by simply considering the aggregated values of the variables such as w_i^A, w_i^B, and $\bar{x}$, our experiments are performed in the batch scenario, as we discuss further in Section 7.5.

7.4 Dataset

In our evaluation study we use a dataset of interleaving experiments obtained from Yandex. For diagnostic purposes, it is useful for a search engine to deploy a constantly running online experiment that compares the current production system with itself ((Kohavi *et al.*, 2012), Section 2). Further, we refer to such an experiment as an A/A comparison. Since we know that the alternatives are equal in this comparison, we want the statistical testing procedure to rarely find statistical differences in this evaluation (i.e. H_0 should be rejected about 1% of the time, when testing is performed on a $p < 0.01$ significance level).

Another source of the experiments are the regular experiments that are deployed to evaluate new search engine improvements. In our evaluation study, we compare the sequential testing approach to a standard scenario, where experiments are deployed for an integer number of weeks. To increase the size of the dataset, we consider the case of the experiments that last for one week. However, as the experiments differ in the expected effect size (detecting smaller differences between A and B require more observations), they also vary in their duration. For this reason, we restrict each experiment to its first 7 days. The ground-truth outcomes used in our evaluation are calculated using the full experimental data. However, the one-step baseline binomial test uses the same 7-day restricted experimental data as is provided for the evaluated sequential tests.

In our dataset, we include data from two interleaving-based A/A experiments in 2014. These two experiments were deployed in two different countries, and contain 108 non-intersecting week-long periods. Further, we sample 336 real-life interleaving experiments that were deployed to evaluate changes in the ranking algorithms. These experiments are the same experiments that were used in Chapter 6, Section 6.8. Among these experiments, we select 109 experiments that lasted for not less than a week and

have a statistically significant outcome, calculated over the full duration of these experiments (binary credit scheme, $p < 0.01$, binomial test). Among these experiments, B outperforms A in 48 experiments.

7.5 Experimental Methodology

The aim of our evaluation experiments is to determine if the sequential tests proposed in Section 7.3 are accurate and increase the evaluation efficiency by decreasing the average time the experiments have to be deployed to get a reliable outcome. We formulate our research questions as follows:

RQ7.1 Is it possible to reduce the deployment time of interleaving experiments by applying the proposed sequential tests?

RQ7.2 Which of the proposed sequential tests achieves the shortest deployment time?

In this section, we discuss the methodology we use to answer these research questions. We split this discussion in two sections. In Section 7.5.1 we introduce the quality metrics we use in our evaluation. Section 7.5.2 describes the evaluation protocol we use.

7.5.1 Metrics

We use six metrics to represent the performance of a sequential test. Our first metric, **Type I error**, represents the probability of a statistical test rejecting the null hypothesis H_0 when it holds:

$$\alpha = P(H_1 \; accepted \,|H_0 \; holds)$$

Generally, we want the Type I error to be low, as each experiment wrongly accepted as successful might result in expensive development, wastes both human and computational resources without improving the search engine.

The second metric we consider is **Type II error**, which measures the probability of accepting the null hypothesis when it does not hold:

$$\beta = P(H_0 \; accepted \,|H_1 \; holds)$$

High values of β indicate that non-equal alternatives A and B are frequently accepted as equal, and this could result in ignoring opportunities to improve a search engine.

The Type II error metric defined as above does not penalise cases when the null hypothesis is correctly rejected, but the preference is inferred incorrectly (e.g. $A \succ B$ is accepted when in reality

$B \succ A$)[48]. Thus we introduce two one-sided accuracy metrics, $Acc_{A \succ B}$ and $Acc_{B \succ A}$:

$$Acc_{A \succ B} = P(accepted\, that\, A \succ B | A \succ B)$$
$$Acc_{B \succ A} = P(accepted\, that\, B \succ A | B \succ A)$$

These metrics are related to the Type II error, however they additionally account for the above discussed cases of the incorrectly inferred preferences.

Mean deployment time, $\mathbb{E}(T)$. This metric is defined as the mean time the experiment is deployed before a sequential testing procedure stops. For the non-sequential one-step test that we use (namely, binomial sign test), the value of this metric is set to the experiment length. For convenience, we measure this metric in days. Generally, it might be more important to stop an experiment where $A \succ B$ than an experiment where $B \succ A$, as in the former case the user experience is degraded. Thus we additionally consider two time-related metrics, which measure the expected duration of the experiments where $A \succ B$ and $B \succ A$. We denote these metrics as $\mathbb{E}(T|A \succ B)$ and $\mathbb{E}(T|B \succ A)$.

Mean relative number of sessions, $\mathbb{E}(\frac{N}{N_0})$. The last metric we use represents the relative number of search interactions required until the experiment is stopped, averaged over all experiments in the dataset. In other words, if an experiment contains N_0 sessions (after restricting to its first 7 days), and the sequential testing procedure stops an experiment after observing only N sessions, the value of the metric is equal to $\frac{N}{N_0}$ on this experiment.

In our evaluation study, for each sequential testing procedure we fix the Type I error probability and the maximum deployment time to be the same. Under these constraints, the baseline one-step approach achieves the minimum Type II error level among all possible rules, as it always "sees" the full experimental data before making a decision. Thus, *our goal is to find a sequential testing procedure that reduces the mean deployment time for the experiments in our dataset as much as possible, while having its Type II error level close to the baseline approach.*

7.5.2 Evaluation Protocol

The A/A experiments, which compare a system with itself are a perfect source of the data to calculate the Type I error probability (i.e. probability of finding a difference when there is none). Indeed, in the A/A experiments, the null hypothesis H_0 definitely holds. Thus, any event where a statistical test detects a difference between the tested alternatives in an A/A experiment, is a Type I error.

Using this observation, we apply the following scheme to measure the Type I error probability for a statistical test. First, we split each of the available A/A experiments in non-overlapping week-long smaller experiments. In the cross-validation process, these parts are split in the training and test sets.

[48] This situation arises since our tests have effectively three outcomes: $A \succ B$, $B \succ A$, and B is not different from A.

The training set is used for learning the stopping thresholds, and the test set is used for calculating the Type I error probability. This probability is calculated as the relative number of experiments where the testing procedure detects a difference between the two compared systems. The initial splitting is repeated in the cross-validation process. We use 25-fold cross-validation.[49]

In the evaluated tests, we set the tolerances for the Type I error to be 0.05. Generally, we want to experiment with lower tolerance values, as this closer resembles the requirements for real experiments. These tolerance levels should be higher than the p-values we use to infer the ground-truth labels, so that the measurements are meaningful. In turn, using low p-values when obtaining the ground-truth labels significantly reduces the sizes of the datasets. Thus we believe that the selected values are reasonable.

To calculate the remaining metrics (Type II error, $Acc_{A \succ B}$, $Acc_{B \succ A}$, $\mathbb{E}(T)$, $\mathbb{E}(\frac{N}{N_0})$), we use the experiments that compare real-life changes in a search engine. As ground-truth labels ($A \succ B$ or $B \succ A$), we use the results of the binomial test ($p < 0.01$).

An alternative approach to calculate Type II errors for both interleaving-based statistical tests is to use a set of experiments, where the tested alternative B is specifically degraded with respect to A. This degradation might be achieved for instance by swapping the documents ranked at the first and the second positions, degrading the quality of snippets, or using an inferior ranking algorithm. The dataset of interleaving experiments for the image search we used in Chapter 6 (Section 6.8) is an example of a dataset that was obtained by applying this approach.

However, building a big dataset of experiments, where the user experience is specifically degraded in different ways, can be unrealistic, since the user experience is deliberately harmed in such experiments. Another concern is that such manually devised degradations cannot be considered as a representative sample of the real-life experiments.

In our evaluation study, we vary the number of the stops used: the stops are performed each day (i.e. 7 stops) and every hour (i.e. $7 \cdot 24 = 168$ stops). While the MaxSPRT-I tests can even be applied at the per-interaction level, we consider scenarios with more stops as impractical for several reasons. Firstly, in the case of the SPRT-based tests the gains cannot be more than the period between two stops due to the design of the test (i.e. more than an hour when comparing cases with stops every hour and per-interaction stops for MaxSPRT-I).[50] On the other hand, to enable frequent testing, one would need to build an elaborated near real-time data delivery system. This is particularly hard, since the data needs

[49] Under the 25-fold cross-validation, 103 A/A experiments (out of 108 available) are used for learning the stopping threshold (Algorithm 7.1). This stopping threshold is set to be $1 - \alpha$ percentile of the SPRT statistic observed on the training set of A/A experiments. Since we set α equal to 0.05, the threshold is determined by 5 experiments with highest values of the SPRT statistic. If a lower number of splits was used, the threshold would be determined by a smaller number of experiments, e.g. in a 5-fold cross-validation only one experiment with the most extreme value of the SPRT statistic would determine the stopping threshold. Clearly, this might result in a high level of noise due to outliers and by using a relatively high number of folds we avoid this.

[50] This observation is also supported by our preliminary experiments.

Table 7.1: The quality metrics of the considered tests, measured on the dataset of interleaving experiments (binary scheme). The groups of values marked with △ are not statistically different from each other and outperform other values in the corresponding columns, $p < 0.01$ (paired t-test across folds). The metric values in bold are the best in the corresponding columns.

Test	# stops	Type I	Type II	$Acc_{B \succ A}$	$Acc_{A \succ B}$	$\mathbb{E}(T \mid B \succ A)$, days	$\mathbb{E}(T \mid A \succ B)$	$\mathbb{E}(T)$	$\mathbb{E}(\frac{N}{N_0})$
Binomial	1	0.026	**0.018**△	**0.96**△	**1.00**△	7.00	7.00	7.00	1.00
OBF-I*	7	0.034	0.055	0.90	0.98	3.77	3.70	3.73	0.53
OBF-I	7	0.060	0.037	0.92	1.00	3.65	3.49	3.56	0.50
MaxSPRT-I-MC	7	**0.000**	0.131	0.85	0.88	3.75	3.55	3.64	0.52
MaxSPRT-I-AA-N	7	0.010	0.110	0.88	0.90	3.47	3.28	3.36	0.48
MaxSPRT-I-AA	7	**0.000**	0.110	0.88	0.90	3.47	3.28	3.37	0.48
OBF-I*	$7 \cdot 24$	0.024	**0.018**△	**0.96**△	**1.00**△	2.94	2.73	2.82	0.39
OBF-I	$7 \cdot 24$	0.058	**0.018**△	**0.96**△	**1.00**△	2.75	2.58	2.66	0.37
MaxSPRT-I-MC	$7 \cdot 24$	0.016	0.087	0.89	0.92	2.29	1.97	2.11	0.29
MaxSPRT-I-AA-N	$7 \cdot 24$	0.052	0.051	0.92	0.96	**1.91**△	**1.74**△	**1.82**△	**0.25**△
MaxSPRT-I-AA	$7 \cdot 24$	0.052	0.046	0.92	0.97	1.92△	**1.74**△	**1.82**△	**0.25**△

to be cleaned from click spam bots and the online experiments typically generate massive data streams. Finally, the O'Brien & Fleming-based test relies on the central limit theorem, hence it requires that sufficiently many interactions occur between stops.

7.6 Results

In this section we discuss the results of our evaluation study (Section 7.6.1) and we perform a visualisation of the decisions of the best-performing test in Section 7.6.2, so as to illustrate its behaviour.

7.6.1 Test Evaluation

Recall that by OBF-I we denote the adaptation of the O'Brien & Fleming test which we discussed in Section 7.3, while OBF-I* denotes the simplified modification of OBF-I that approximates the variance by the unity. MaxSPRT-I-MC is the MaxSPRT test with its $\bar{L}$ threshold trained by the Monte-Carlo approach. In contrast, MaxSPRT-I-AA corresponds to the MaxSPRT test with its $\bar{L}$ threshold trained on the dataset of A/A experiments by Algorithm 7.1. Binomial denotes a one-step testing procedure that runs the Binomial test at the end of an interleaving experiment. The five sequential tests we evaluate (OBF-I, OBF-I*, MaxSPRT-I-MC, MaxSPRT-I-AA, and MaxSPRT-I-AA-N) were introduced in Section 7.3. The metrics we use to evaluate these tests are defined in Section 6.10.2.

In Table 7.1, we report the results of the evaluation of the sequential testing rules on the dataset of the interleaving experiments. The case of 7 stops corresponds to the scenario where interim stops are performed every 24 hours, and $7 \cdot 24$ indicate the scenario where interim analysis is performed every hour.

On analysing these results we firstly notice that the Type I error levels measured for all the considered testing rules are close to the tolerance level we set in the threshold learning process (Section 7.3), namely 0.05. The observed deviations might be caused by the limited size of the dataset we use.

Further, on analysing the Type II error metric reported in Table 7.1, we notice that the values of this metric are close within each family of the tests: in the range of $0.02 - 0.05$ for the OBF-I-based tests and in the range of $0.05 - 0.11$ for the MaxSPRT-based tests. The highest level of Type II errors is demonstrated by the MaxSPRT-I-MC test (0.131 and 0.087, for the cases with 7 and $7 \cdot 24$ stops, respectively). In Table 7.1, the lowest Type II error level (0.018) is achieved by the OBF-I* and OBF-I tests in the scenario with $7 \cdot 24$ stops. Similarly, they demonstrate the highest $Acc_{A \succ B}$ and $Acc_{B \succ A}$ metrics. In particular, the $Acc_{B \succ A}$ metric of 0.96 and the $Acc_{A \succ B}$ of 1.00 are achieved when $7 \cdot 24$ stops are used.

As can be seen from Table 7.1, all the evaluated sequential tests achieve considerable improvements over the standard 7-day scenario and this observation answers RQ7.1. Among the tests that use 7 stops, on average, OBF-I* stops the experiments later than other tests (e.g. 3.73 OBF-I* vs. 3.37 MaxSPRT-I, 7 stops). The shortest mean time (3.36) is demonstrated by the MaxSPRT-I-AA-N test.

On comparing the scenarios with 7 and with $7 \cdot 24$ stops, we firstly notice that the MaxSPRT-I-MC, MaxSPRT-I-AA, and MaxSPRT-I-AA-N tests greatly benefit from using additional stops. Indeed, the mean time is reduced for the MaxSPRT-I-MC test from 3.64 to 2.11. Similarly, MaxSPRT-I-AA and MaxSPRT-I-AA-N have improved their mean experiment running time from 3.37 and 3.36, respectively, to 1.82, and achieved the best performance. This behaviour is intuitive: with more stops available, there is more potential to stop earlier.

The OBF-I and OBF-I* tests demonstrate smaller improvements. For instance, OBF-I increased the mean deployment time from 3.56 to 2.66.

From Table 7.1, we observe that MaxSPRT-I-AA and MaxSPRT-AA-N have the shortest mean deployment time, both among the tests with 7 stops (3.37 and 3.36, respectively), and among the tests with $7 \cdot 24$ stops (1.82). When 7 stops are used, OBF-I has a relatively close performance (3.56), but underperforms in the case of $7 \cdot 24$ stops. Overall, MaxSPRT-I-AA and MaxSPRT-AA-N have almost identical results, except for the Type II error and Accuracy metrics in the case of $7 \cdot 24$ stops. It is generally expected that MaxSPRT-AA-N performs slightly worse, as it uses a somewhat coarse normal approximation to the data. However, the same approximation allows it to handle non-binary interaction outcomes.

An interesting observation is that the difference in the mean deployment times for OBF-I and OBF-I* are relatively close (not more than 0.17 days or 4 hours in the scenario with 7 stops). However, the

difference between MaxSPRT-I-AA and MaxSPRT-I-MC is bigger (0.29 days ≈ 7 hours maximum). In all scenarios MaxSPRT-I-AA outperforms MaxSPRT-I-MC, indicating that replacing the Monte-Carlo threshold estimate with the threshold learned from the A/A tests improves the test's performance.

We also observe that the relative improvements measures by the $\mathbb{E}(T)$ metric are well aligned with the improvements measured by the $\mathbb{E}(\frac{N}{N_0})$ metric (e.g. MaxSPRT-I-AA with $7 \cdot 24$ stops reduces the mean deployment time by 74%, and uses only 0.25 of the available sessions).

We conclude that in the case of interleaving, the MaxSPRT-I-AA and MaxSPRT-I-AA-N tests with $7 \cdot 24$ stops achieve the smallest deployment time and this observation answers RQ7.2. In comparison to the standard 7-day scenario, the improvement corresponds to 74% increase in the efficiency (1.82 vs. 7.00 days).

7.6.2 Visualisation

We illustrate the best-performing MaxSPRT-I-AA test as follows. First, we sample a random subset of experiments, including experiments with A outperforming B (according to the ground-truth labels), B outperforming A, and A/A experiments. Second, for each of these experiments, at each stop i we calculate the log-likelihood ratio L_i multiplied by the sign of the current estimate of difference between A and B. More specifically, we multiply the log-likelihood (Equation (7.6)) by the sign of the current estimate of $(\hat{p}_1^i - \frac{1}{2})$:

$$sign\left(\hat{p}_1^i - \frac{1}{2}\right) \cdot L_i \tag{7.9}$$

By definition, the absolute value of Equation (7.9) is equal to the log-likelihood ratio L_i, and its sign indicate which system tends to be better according to the observed data (e.g. if (7.9) is positive, then $B \succ A$, and vice-versa). As a result, whenever the value of Equation (7.9) leaves the corresponding interval $[-\bar{L}, +\bar{L}]$, the experiment is terminated and a decision is made (e.g. $B \succ A$ if the upper boundary is touched).

We report our obtained results in Figure 7.1. The green and red lines correspond to the experiments that are labelled as $B \succ A$ and $A \succ B$ according to the ground-truth labels. The black lines correspond to the sampled A/A experiments. The horizontal dashed lines indicate the boundaries of the intervals $[-\bar{L}, +\bar{L}]$.

From Figure 7.1 we observe that despite some fluctuations, the likelihood ratios for the sampled A/A experiments are well within the boundaries of the decision interval at each of the stops. Next, for most of the interleaving experiments with A outperforming B, the statistic (7.9) falls towards the lower bound $(-\bar{L})$. In contrast, several experiments with $B \succ A$ have their statistic reaching the wrong boundary.

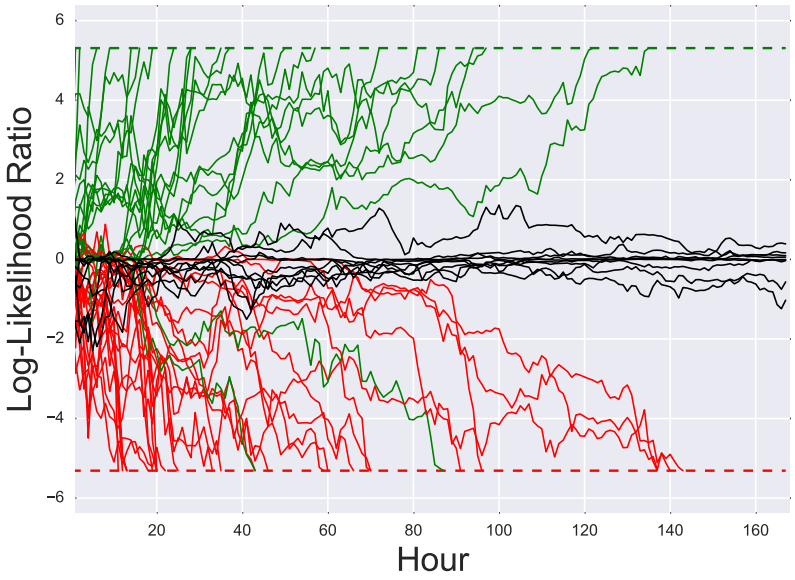


Figure 7.1: Illustration of the MaxSPRT-I-AA test, binary scheme. Green and red lines correspond to the experiments with $B \succ A$ and $A \succ B$ ground-truth labels, respectively. Black lines correspond to A/A experiments. The horizontal dashed lines denote the threshold values for accepting $B \succ A$ (green) and $A \succ B$ (red).

These cases result in decreasing the $Acc_{B \succ A}$ value. This observation agrees with the results reported in Table 7.1. Indeed, the $Acc_{B \succ A}$ metric values are lower than $Acc_{A \succ B}$ in every row of these tables. Thus, in a random sample it is more likely to meet errors of wrongly rejecting $B \succ A$ than the opposite case of rejecting $A \succ B$.

Overall, from our experiments we observe that by using sequential testing procedures considerable gains can be obtained in the mean time experiments are deployed and this observation answers RQ7.1. At the same time, we notice that this improvement is obtained without significant degradations in other metrics, such as Type I, Type II errors, $Acc_{A \succ B}$, and $Acc_{B \succ A}$. A reduction in the execution time of 74% can be achieved by using the MaxSPRT-I-AA and MaxSPRT-I-AA-N tests with $7 \cdot 24$ stops, and this is the shortest mean deployment time we observed in our experiments in this section. This answers RQ7.2.

7.7 Conclusions

In this chapter we addressed an important problem of increasing the efficiency of the online evaluation experiments. In particular, we studied how sequential testing procedures can be adapted to reduce the

time online evaluation experiments require. These procedures are designed so that they can stop online experiments when the observed data is sufficient to make a reliable conclusion about the experiment's outcome.

We proposed a modification of the O'Brien & Fleming group sequential test that can be applied to interleaving evaluation. Further, we described an approach to improve the MaxSPRT test's performance by adjusting its stopping threshold on the dataset of A/A experiments.

In our evaluation study we used the same dataset as used in Chapter 6. Of note, our dataset consists of 109 experiments lasting at least 7 days long and have a statistically significant difference between A and B, which is larger than other datasets used in the existing literature. Our study demonstrates that by using the sequential testing procedures, a marked reduction in the duration of the experiments can be achieved, without significant losses in other metrics, such as Type I and Type II error probabilities. The maximal reduction in the mean deployment time over a standard 7 day one-stop scenario on the dataset of interleaving experiments reaches 74% by using the MaxSPRT-I-AA and MaxSPRT-I-AA-N tests, that examine the experiment data every hour. Clearly, such an improvement might have a considerable impact on the evaluation pipeline efficiency.

This improvement can be illustrated as follows. If each experiment is deployed for 7 days, 109 experiments in our dataset will require $7 \cdot 109 = 763$ experiment-days of evaluation. Assuming the deployment time can be reduced by 74%, these experiments can be evaluated in approximately 200 days. Roughly, this indicates that we are able to evaluate three datasets of interleaving experiments of the same size in the same time period.

Overall, we notice that our work in this chapter support the statement of this thesis (Section 1.3). Indeed, by using historical interaction data recorded in A/A experiments, we build sequential tests that are capable of stopping online interleaving experiments earlier and thus increasing the evaluation pipeline efficiency. This work corresponds to the third point of the our roadmap for improving the evaluation pipeline, discussed in Section 3.5.

In this chapter we discussed how sequential testing procedures can be applied in the online evaluation to reduce the duration of the online experiments. In Chapter 6, we proposed the Generalised Team Draft framework that optimises interleaving sensitivity so that experiments can be deployed for a shorter time period. Both these approaches aim to improve the efficiency of the online evaluation step, and in the next chapter we demonstrate that these two approaches are complimentary and can be used in conjunction. Furthermore, we demonstrate that by using them in combination even higher improvements in the online efficiency can be achieved.

Chapter 8

Early Stopping of Sensitive Interleaving Experiments

8.1 Introduction

In Section 2.4, we noted that the online evaluation step is one of the most time-consuming steps, as online experiments typically take a period lasting from several days up to several weeks. Hence, improving the efficiency of the online evaluation step would have a dramatic influence on the efficiency of the entire evaluation pipeline.

Previously, we addressed the problem of increasing the interleaving efficiency from two perspectives. In Chapter 6, we proposed the Generalised Team Draft framework that increases the interleaving sensitivity by a data-driven optimisation of the interleaving policy (how often a particular interleaved result page is shown) and the click weighting scheme (how important each click is). In Chapter 7, we discussed how sequential testing procedures can be used to reduce the average duration of online experiments by stopping them earlier.

Both these approaches used historical interaction data to optimise the parameters of interleaving and sequential tests, and achieved marked improvements in the evaluation pipeline efficiency, thus supporting the statement of this thesis (Section 1.3). A further question therefore arises: *can these two approaches be combined together in a single evaluation mechanism such that cumulative improvements in the efficiency of the online evaluation through interleaving can be achieved?* In this chapter we aim to experimentally answer this question.

From our experimental study of the sensitive interleaving schemes in Section 6.11, we observed that the highest sensitivity is achieved by the interleaving algorithms which assign real-valued scores to the user clicks, such as Generalised Team Draft, defined in Section 6.4. In Chapter 7, we studied

two families of sequential tests for interleaving, described in detail in Section 7.3: (a) tests based on the O'Brien & Fleming test (O'Brien & Fleming, 1979), i.e. OBF-I and OBF-I*, and (b) tests based on the MaxSPRT test (Kulldorff *et al.*, 2011), i.e. MaxSPRT-I-MC, MaxSPRT-I-AA, MaxSPRT-I-AA-N. Among these tests, OBF-I and MaxSPRT-I-AA-N achieve the shortest mean deployment time in the corresponding families[51] and can handle real-valued scores for user clicks. Hence, to reduce the number of possible combinations of scoring schemes and sequential tests while keeping the most interesting ones, in this chapter we restrict our study to these two sequential tests.

The chapter is organised as follows. We start with Section 8.2, where we discuss how our proposed sequential tests can be adopted to leverage the stratified interleaving outcome estimator used in Generalised Team Draft. In Section 8.3, we introduce the dataset used in this chapter. In Section 8.4 we describe the experimental methodology we use. The obtained results are reported and discussed in Section 8.5. In Section 8.6 we provide concluding remarks.

8.2 Stratified Sequential Testing

The Generalised Team Draft framework proposed in Chapter 6 (Section 6.4) assumes that the interleaving outcomes are stratified w.r.t. team combinations shown to the users. In order to meet this expectation, we need to adapt the statistical testing procedures used to test the significance of the observations.

The core sample statistics controlled by both the OBF-I and MaxSPRT-I-AA-N tests are the mean score and the score variance statistics. Assuming that the stratified estimator we proposed in Section 6.6 (Equation (6.15), page 104) is used, we change how these statistics are calculated. The stratified mean score is calculated as a weighted combination of per-strata means:

$$\hat{\Delta}_s = \sum_i \pi_i \cdot \frac{1}{|\mathbb{A}_i|} \sum_{a \in \mathbb{A}_i} S(a) \tag{8.1}$$

Again, i iterates from 1 to 32, enumerating all allowed team combinations on the first page of ten results from *abababababa* to *bababababa*; $\mathbb{A}_i$ is the set of interactions with the ith team combination shown to the users; π_i is the probability of showing the ith team combination; $S(a)$ is the score associated for the interaction a. For example, $S(a)$ can be calculated as defined by the Generalised Team Draft in Section 6.4.

The sequential testing procedures we study operate with the accumulated scores aggregated between stops. We denote the mean score of the interactions in the ith stratum, performed before the jth stop, as

[51] MaxSPRT-I-AA-N and MaxSPRT-I-AA with $7 \cdot 24$ stops demonstrate the shortest mean deployment time among all studied tests (see Table 7.1, page 136).

$\bar{x}_i^j$, calculated as follows:

$$\bar{x}_i^j = \frac{1}{|\mathbb{A}_i^j|} \sum_{a \in \mathbb{A}_i^j} S(a) \tag{8.2}$$

where $\mathbb{A}_i^j$ is the set of user interactions that took place before the jth stop and has the ith team combination demonstrated to the users.

Under this notation, at the jth stop, the mean stratified score can be found as follows:

$$\Delta_s^j = \sum_i \pi_i \cdot \bar{x}_i^j \tag{8.3}$$

The variance of the stratified estimator Δ_s^j calculated at the jth stop is:

$$D[\Delta_s^j] = \sum_i \pi_i^2 \cdot D[\bar{x}_i^j] = \sum_i \frac{\pi_i^2}{N_i^j} \cdot D_i^j[x] = \frac{1}{N^j} \sum_i \pi_i \cdot D_i^j[x] \tag{8.4}$$

where $N_i^j = |\mathbb{A}_i^j|$ is the number of interactions in the ith stratum before the jth stop, and N^j is the total number of interactions that occurred before the jth stop. Similarly, $D_i^j[x]$ is the variance of the outcomes within the ith strata, calculated based on observations before the jth stop:

$$D_i^j = \frac{1}{N_i^j - 1} \sum_{a \in \mathbb{A}_i^j} \left(S(a) - \bar{x}_i^j \right)^2 \tag{8.5}$$

Having Equations (8.3) and (8.4) in mind, we can specify the sequential testing criteria for the OBF-I and for the MaxSPRT-I-AA-N tests.

OBF-I At the jth stop, the OBF-I test statistic can be calculated as follows:

$$O_j = j \cdot \frac{\left(\Delta_s^j\right)^2}{D[\Delta_s^j]} = j \cdot N^j \frac{\left(\sum_i \pi_i \cdot \bar{x}_i^j\right)^2}{\sum_i \pi_i \cdot D_i^j[x]} \tag{8.6}$$

Again, just as in the "vanilla" OBF-I, the O_j statistic is compared to a fixed threshold, $\bar{O}$. This threshold is learned by means of Monte-Carlo simulations, as described in Section 7.3.1.

MaxSPRT-I-AA-N Similarly, the stratified MaxSPRT statistic used in the MaxSPRT-I-AA-N test is calculated as follows:

$$L_i = \log \frac{\phi(\Delta_s^j; \Delta_s^j, D[\Delta_s^j])}{\phi(\Delta_s^j; 0, D[\Delta_s^j])} \tag{8.7}$$

where the current stratified mean score Δ_s^j and its variance $D[\Delta_s^j]$ are defined in Equations (8.3) & (8.4), respectively. The function $\phi(x; \mu, \sigma^2)$ is the probability density function of the normal distribution with some mean μ and variance σ^2. We set the stopping threshold $\bar{L}$ on L_j by the training procedure on the A/A experiments, as described by Algorithm 7.1 in Section 7.3.2.

As we discussed earlier in Section 6.6, it is incorrect to ignore interactions without clicks in combination with the stratified estimator Equation (8.3). To keep the experimental setup uniform for all

Table 8.1: Datasets statistics.

# exp.	A $\succ$ B	mean # interactions	median # interaction
131	66	593K	328K

combinations of scoring schemes and sequential tests, in this chapter we do not ignore the interactions without clicks, and assign them with the zero score $S(a)$.

8.3 Dataset

As a foundation for our experimental study in this chapter, we used the same 336 real-life Team Draft interleaving experiments that were used in Chapters 6 and 7. To alleviate the consequences of the down-sampling evaluation scenario we use in this chapter (discussed in Section 8.4.2.1) that reduces the number of experiments with statistically significant outcome, we added 42 interleaving experiments that were performed in 2015, after the span of the earlier used dataset was finished. From these 378 experiments, we selected 131 experiments that were deployed for at least a week and have a statistically significant difference between A and B (deduped binary click scheme, binomial sign test, $p < 0.01$). The descriptive statistics of the dataset are provided in Table 8.1.

Next, we used the data from two interleaving-based A/A experiments deployed in 2014. These two experiments were deployed in two different countries, and contain 108 non-intersecting week-long periods. These are the same A/A interleaving experiments that were described in Section 7.4 and were used to adjust the stopping thresholds of the MaxSPRT-I tests and to evaluate the Type I errors in Chapter 7.

8.4 Experimental Methodology

In Chapter 6 (Section 6.11), we compared the sensitivity of seven interleaving scoring schemes. Five of these schemes are heuristic schemes, and two schemes optimise their parameters against a set of earlier performed interleaving experiments. We observed that these machine-learned schemes are more sensitive. In particular, Generalised Team Draft demonstrated the highest sensitivity in our study. Further, in Chapter 7, we studied sequential tests that come from two families: the tests based on the O'Brien & Fleming test (O'Brien & Fleming, 1979), and those based on the MaxSPRT test (Kulldorff *et al.*, 2011). From our experiments in Section 7.6, we observed that the MaxSPRT-based tests achieved the shortest mean deployment time.

In this chapter we aim to understand if these two approaches to improve the interleaving efficiency can be combined together to achieve even further gains in efficiency. We split the broad question stated earlier in this chapter in Section 8.1 in three more specific research questions that we aim to answer:

RQ8.1 How does the sensitivity of the interleaving scoring scheme influence the mean deployment time, if we fix a sequential test?

RQ8.2 With an interleaving scoring scheme fixed, how do the sequential tests compare in terms of the mean deployment time?

RQ8.3 Which of the combinations of the interleaving scoring schemes and sequential tests is the most efficient (i.e. achieves the shortest mean deployment time)?

To answer these questions, we firstly review the metrics we use in Section 8.4.1. In Section 8.4.2 we discuss the evaluation protocol.

8.4.1 Metrics

In this chapter we adopt the metrics from Chapter 7 (Section 7.5.1). To recap, these metrics are the following:

1. Type I error probability,
2. Type II error probability,
3. Accuracy metrics *AccAB* and *AccBA*,
4. Mean deployment time, $\mathbb{E}(T)$.

By definition, Type I error measures how often a particular combination of the statistical test and the scoring scheme finds a difference when there is none. In contrast, the Type II error probability denotes how often a particular combination cannot find such a difference when one is present. The *AccAB* and *AccBA* metrics reflect the ratio of the experiments with the correctly detected preference relation (e.g. each time the $A \succ B$ preference is correctly detected, the $AccAB$ metric is increased). Finally, the mean deployment time represents an average time an experiment is deployed for, and directly measures the efficiency of the online evaluation step of the evaluation pipeline (Chapter 3, Section 3.3). In Chapter 7, we also used the $\mathbb{E}(\frac{N}{N_0})$ metric that represents the mean ratio of the experiment interactions observed before the experiment was stopped. In this chapter we exclude it, since our experiments in Chapter 7 demonstrated that it is highly correlated with the mean deployment time metric.

Clearly, the mean deployment time metric is strongly related to the efficiency of the online evaluation step. A smaller deployment time implies that more online experiments can be deployed in a unit of time, thus directly indicating a higher efficiency. Like in Chapter 7, our goal in the online evaluation efficiency optimisation is to achieve the shortest deployment time possible while preserving reasonable levels of Type I and Type II errors.

8.4.2 Evaluation Protocol

The Team Draft modifications we have studied in Chapter 6 can be split in two groups. The modifications in the first group use the default uniform interleaving policy, so that each possible team combination is equally likely, and only the click weights are changed (Section 6.10.1). In the second group, the interleaving policy is an optimised parameter and, in general, an arbitrary multinomial distribution subject to the unbiasedness constraint (Section 6.4, requirement R1) can be used as a policy. The Generalised Team Draft interleaving framework we proposed in Chapter 6 falls in the second group.

Since only a dataset of interleaving experiments that were performed under the uniform policy is available to us, we split our experimental evaluation in two major steps. In the first step, we perform an evaluation study using the Team Draft modifications that assume the uniform policy. We discuss how such an evaluation can be performed in Section 8.4.2.1. In the second step, we study how the combinations of the sequential tests with Generalised Team Draft compare with the combinations of the sequential tests with the best-performing scoring scheme from the first step. Since the available experiments were performed under the uniform policy, a fair comparison of the combinations of Generalised Team Draft with sequential tests against the combinations with Team Draft modifications that use uniform policy is a hard task. We discuss how it can be performed in Section 8.4.2.2.

8.4.2.1 Uniform Policy Evaluation

Both sensitive scoring schemes introduced in Chapter 6 (e.g. Generalised Team Draft, Section 6.4) and sequential testing procedures from Chapter 7 (Section 7.3) can learn their parameters from the data. For instance, the scoring scheme L_z, introduced by Yue *et al.* (2010), optimises the scoring scheme parameters against the labelled ($A \succ B$ or $B \succ A$) dataset of experiments. In turn, the MaxSPRT-I-AA-N test optimises its stopping threshold on a dataset of A/A experiments so that the required Type I error level is met. Clearly, this stopping threshold depends on the actual distribution of the interactions scores defined by the underlying scoring scheme, thus it depends on the scoring scheme learned.

To resolve this dependency, we perform these training procedures sequentially. First, we optimise the scoring scheme using a training set of interleaving experiments. Next, we adjust the stopping thresh-

Input: A set of scoring schemes $\mathbb{C}$ and a set of statistical tests $\mathbb{T}$ to be evaluated; a set of interleaving experiments $\mathbb{E}_{AB}$ and A/A experiments $\mathbb{E}_{AA}$; the required Type I error level α.

Output: Quality metrics: TypeI and Type II error levels, mean deployment time Time; accuracy metrics AccAB and AccBA.

A cross-validation loop over the interleaving experiments used to optimise interleaving sensitivity
for $\mathbb{E}_{AB}^{train}, \mathbb{E}_{AB}^{test} \leftarrow CrossValidate(data = \mathbb{E}_{AB}, folds = 15)$ **do**
  for $c \in \mathbb{C}$ **do**
    We optimise the scoring scheme c based on the training subset of the experiments $\mathbb{E}_{AB}^{train}$
    c.train(trainExp)
    We split A/A experiments to optimise the stopping thresholds of the statistical tests using $\mathbb{E}_{AA}^{train}$, and to measure the Type I error levels using $\mathbb{E}_{AA}^{test}$
    for $\mathbb{E}_{AA}^{train}, \mathbb{E}_{AA}^{test} \leftarrow CrossValidate(data = \mathbb{E}_{AA}, folds = 15)$ **do**
      Get the per-interactions scores according to the optimised scoring scheme c
      $\mathbb{E}_{AA}^{train} \leftarrow c.apply(\mathbb{E}_{AA}^{train})$
      $\mathbb{E}_{AA}^{test} \leftarrow c.apply(\mathbb{E}_{AA}^{test})$
      Adjusting the stopping thresholds on the A/A experiments
      for $t \in \mathbb{T}$ **do**
        t.train($\mathbb{E}_{AA}^{train}$, α)
        TypeI, TypeII, AccAB, AccBA, Time $\leftarrow$ CalcMetrics($\mathbb{E}_{AB}^{test}$, $\mathbb{E}_{AA}^{test}$, t)
      end
    end
  end
  return (TypeI, TypeII, AccAB, AccBA, Time) averaged over all cross-validation splits
end

Algorithm 8.1: Evaluating the combination of the interleaving sensitivity optimisation methods and sequential testing procedures.

old for the sequential test using the distribution of the scores generated by this pre-trained scoring scheme on a training set of A/A experiments. The hold-out test sets of experiments are then used to evaluate the performance of the combination of the sequential test and the scoring scheme.

We report this evaluation procedure more formally in Algorithm 8.1. Algorithm 8.1 uses two nested cross-validation loops. The first cross-validation loop splits the set of experiments $\mathbb{E}_{AB}$ in two parts that are used to train the scoring scheme and to evaluate the Type II, AccAB, AccBA, $\mathbb{E}(T)$ metrics. The second cross-validation loop splits the set of A/A experiments $\mathbb{E}_{AA}$ so that the training part is used to adjust the stopping threshold, while the second part is used to evaluate the Type I error metric.

8.4.2.2 Non-uniform Policy Evaluation

To illustrate the problems we face when evaluating an interleaving algorithm with a non-uniform policy using a dataset of the experiments performed under the uniform policy, let us consider the following

Input: A set of experiment interactions $\mathbb{A}$ performed under the uniform policy; the target policy $\boldsymbol{\pi}^n$.

Output: Two down-sampled experiments of equal size with the uniform and target policies: $\mathbb{A}^u$, $\mathbb{A}^n$.

By N we denote the number of interactions in the experiment after the down-sampling. The pseudo code below guarantees that N is the maximal number such that there are at least $\pi_i^n \cdot N$ interactions for each stratum to sample from

$N \leftarrow |\mathbb{A}|$
for $i = 1..32$ **do**
  $N_i \leftarrow |\mathbb{A}_i|$
  if $\pi_i^n > 0$ **then**
    $N \leftarrow \min(N, N_i/\pi_i^n)$
  end
end

The down-sampled experiment with the uniform policy is obtained by simply sampling N interactions

$\mathbb{A}_u \leftarrow UniformSample(from = \mathbb{A}, count = N)$

The down-sampled experiment with the non-uniform policy is obtained by sampling strata separately such that the ith strata contains $\pi_i^n \cdot N$ interactions

for $i = 1..32$ **do**
  $\mathbb{A}_i^n \leftarrow UniformSample(from = \mathbb{A}_i, count = \pi_i^n \cdot N)$
end
$\mathbb{A}^n \leftarrow \bigcup_i \mathbb{A}_i^n$

Algorithm 8.2: Down-sampling an interleaving experiment. As a result of running this algorithm, two equal-sized interleaving experiments are generated, one of them contains interactions sampled under the uniform policy, and the second contains the interactions under the target policy.

example. Assume we have an experiment that contains a set of user interactions $\mathbb{A}$ performed under the uniform policy. Once we want to use these interactions to simulate an experiment with a non-uniform policy $\boldsymbol{\pi}^n$ that has non-zero probabilities only for two team combinations T_1 and T_{32} (e.g. the policy obtained by Generalised Team Draft in Section 6.11.3 when trained on the full dataset), we cannot re-use all sessions where one of the remaining 30 team combinations was demonstrated. In other words, only $2 \cdot \frac{1}{32}$ of the data can be used, thus effectively reducing the size of each experiment in the dataset down to $\frac{1}{16}$ of its original size. After such a down-sampling, a two-week long experiment will have less interactions than a single-day experiment without down-sampling. In order to compare Generalised Team Draft to other Team Draft modifications fairly, we need to ensure that for each of the experiments used in the evaluation, all scoring schemes (with either a uniform or a non-uniform policy) use the same number of interactions. We use two different evaluation scenarios to guarantee that.

Down-sampling To illustrate the first scenario, consider the above discussed example of comparing two policies, the uniform policy $\boldsymbol{\pi}^u = \frac{1}{32}(1, 1, ..., 1)^T$, and the non-uniform policy $\boldsymbol{\pi}^n$ with two

non-zero elements, T_1 and T_{32}, $\boldsymbol{\pi}^n = \frac{1}{2}(1, 0, ..., 0, 1)^T$. Given an experiment[52] associated with a set of interactions $\mathbb{A}$, we can obtain an experiment "performed" under $\boldsymbol{\pi}^n$ by selecting a subset of interactions $\mathbb{A}^n \subset \mathbb{A}$ in such a way that this subset has equal number of interactions with team combinations L_1 and L_{32} ($|\mathbb{A}^n_1| = |\mathbb{A}^n_{32}|$), and no other team combinations present. Similarly, we can generate an experiment with the uniform policy by selecting a subset of interactions $\mathbb{A}^u \subset \mathbb{A}$ such that each team combination is equally represented in $\mathbb{A}^u$ ($|\mathbb{A}^u_1| = |\mathbb{A}^u_2| = ...$). To ensure fair comparison between generated experiments, we need to guarantee that we obtain experiments of equal number of interactions. This requirement can be expressed as follows:

$$|\mathbb{A}^u| = |\mathbb{A}^n| \tag{8.8}$$

At the same time, for each of the resulting experiments, the frequency of the team combinations must follow the corresponding policy:

$$\forall i : \frac{|\mathbb{A}^u_i|}{|\mathbb{A}^u|} = \pi^u_i, \quad \frac{|\mathbb{A}^n_i|}{|\mathbb{A}^n|} = \pi^n_i \tag{8.9}$$

Finally, as we re-use interactions from the original experiment (i.e. $\mathbb{A}^u_i \subset \mathbb{A}_i$ and $\mathbb{A}^n_i \subset \mathbb{A}_i$), for each possible result page, there must be enough interactions to select from. Hence, the following inequalities must be satisfied:

$$\forall i : |\mathbb{A}^u_i| = \pi^u_i \cdot |\mathbb{A}^u| \leq |\mathbb{A}_i|, \quad |\mathbb{A}^n_i| = \pi^n_i \cdot |\mathbb{A}^n| \leq |\mathbb{A}_i| \tag{8.10}$$

Overall, our goal is to select two sets of interactions $\mathbb{A}^u$ and $\mathbb{A}^n$ from the initial set $\mathbb{A}$ such that (a) the conditions specified by Equations (8.8)-(8.10) are met, (b) $\mathbb{A}^u$ and $\mathbb{A}^n$ have the highest possible cardinality, as this allows us to achieve a higher statistical significance of the outcomes of the resulting experiments.

Algorithm 8.2 describes how these goals can be achieved. In the first lines, Algorithm 8.2 finds the number of interactions in the sub-sampled experiment with a non-uniform policy, $N = |\mathbb{A}^n|$. This number N should satisfy the condition specified by Equation (8.10): for the ith team combination with the non-zero probability π_i there should be enough interactions $\mathbb{A}_i$ with ith team combination in the source experiment to sample from, i.e. $|\mathbb{A}_i| \geq N \cdot \pi_i$. After N is determined, we proceed to down-sampling the resulting experiments. To build an experiment with the uniform policy and N interactions, we uniformly at random select N interactions from the source experiment and obtain a sample $\mathbb{A}^u$. The experiment with a non-uniform policy $\boldsymbol{\pi}^n$ is obtained by sampling $N \cdot \pi^n_i$ interactions from each stratum $\mathbb{A}_i$. Since we selected N such that $|\mathbb{A}_i| \geq N \cdot \pi_i$ for all strata i, it is guaranteed that we always have sufficient interactions to sample from.

[52] We assume that during this experiment all possible team combinations were shown to the users.

The resulting evaluation procedure only slightly differs from the one used in the case of uniform policy that is described in Algorithm 8.1. The only modification is that in the outer cross-validation loop that splits $\mathbb{E}_{AB}$, Generalised Team Draft is always trained first. The optimised interleaving policy it finds is then used to down-sample all experiments used further: the testing set $\mathbb{E}_{AB}^{test}$ and all A/A experiments $\mathbb{E}_{AA}$. Such a process ensures that all combinations are trained and tested using an equal number of interactions. In order to reduce the additional randomness due to the down-sampling, we repeat it 16 times for each combinations of splits of $\mathbb{E}_{AB}$ and $\mathbb{E}_{AA}$ and average the values of the metrics.

Simulation To motivate the second evaluation scenario, remember that at the jth interim stop, both MaxSPRT-based test and O'Brien & Fleming test make a decision relying on the following statistics: per-strata mean scores $\bar{x}_i^j$, per-strata variances $D_i^j[x]$, and the number of interactions occurred in each strata N_i^j (Section 8.2, Equations (8.3)-(8.7)).

Suppose that we are comparing Generalised Team Draft to another scoring scheme that assumes the uniform policy, both in combination with the same sequential test. Under the second scenario, when calculating the test statistics for the sequential tests (Equations (8.7) & (8.6)), we use the per-strata mean scores and the per-strata variances calculated using the full experimental data (i.e. they are equal for all interim stops). However, we vary the number of sessions occurred ($N^j = \sum_i N_i^j$) and set it to be equal for the compared interleaving methods. At each interim stop, N_j is set to the same value as in the down-sampling scenario, by running Algorithm 8.2. In other words, in the simulation scenario, for each scoring scheme we fix the mean and variance to their converged values and only vary the number of interactions observed before the current stop.

On one hand, such a simulation favours the scoring scheme with the uniform policy, as its mean and covariance matrix are calculated using a larger amount of data. On the other hand, assuming that the means and the variances of both scoring schemes converge at the end of the experiment, such an evaluation allows us to measure the improvements in the sensitivity in terms of the stopping times under a particular sequential test and addresses the problems caused by down-sampling.

Overall, we note that such a measurement is clearly a simulation as it removes some intrinsic noise from the test statistics obtained on early stops and uses information from the "future" of the experiment. In this scenario, we use a modification of the MaxSPRT-I-AA-N test that is trained by using Monte-Carlo simulations, very similar to the OBF-I test (Section 7.3.1), as training the stopping thresholds of MaxSPRT-I-AA-N on such simulated data is not reasonable. We refer to this test as MaxSPRT-I-MC-N.

To summarise, in this section, we discussed two approaches to compare Generalised Team Draft with a non-uniform policy against other Team Draft modifications that assume the uniform interleaving policy, in combinations with sequential testing procedures. The first approach relies on down-sampling

each experiment in the dataset into two smaller experiments with equal number of interactions, such that one of them has the uniform distribution of the team combinations, and the second one has the distribution specified by Generalised Team Draft. Under the second approach, we set the mean and the variance of the interaction scores for both schemes, and vary the number of interactions N^j observed before the jth stop of the sequential test. This number is equal to the number of interactions we would obtain before the jth stop after down-sampling.

8.5 Results

In this section, we adopt the notation used before in Chapter 6 (Section 6.11) and in Chapter 7 (Section 7.6). By *Binary*, *Linear*, and *Deduped* we denote the heuristic interleaving scoring schemes that are studied in Chapter 6 and are defined in Section 6.10.1. By L_z we denote the machine-learned click scoring scheme that optimises z-score (Yue *et al.*, 2010). This scoring scheme was used also in Chapter 6 and defined in Section 6.10.1.

Similarly, F_z denotes the Generalised Team Draft variant that optimises z-score. In contrast to Generalised Team Draft, L_z assumes a uniform interleaving policy and does not use stratification. While training both L_z and F_z we use the 24-dimensional click feature representation as used in Chapter 6 (Table 6.3, page 110).

The OBF-I and MaxSPRT-I-AA-N sequential tests were firstly introduced in Sections 7.3.1 and 7.3.2, respectively. Their stratified modifications, which can be used in combination with Generalised Team Draft are described in Section 8.2. Throughout this section, we use the same parameters for the sequential tests in Chapter 7. We set the required Type I error level to $\alpha = 0.05$. All sequential tests are allowed to stop experiments every hour, so that up to $7 \cdot 24 = 168$ stops are performed.

We split the discussion of the experimental results in two parts. In Section 8.5.1 we discuss our experiments under the uniform policy, and these experiments are performed according to the methodology discussed in Section 8.4.2.1. In Section 8.5.2, we proceed to the experiments with a non-uniform policy, as described in Section 8.4.2.2.

8.5.1 Uniform Policy

In Table 8.2 we report the evaluation results for the interleaving scoring schemes that assume the uniform Team Draft policy. First, we notice that the measured Type I errors are roughly equal to the one we set (0.05) for most of the used combinations of the scoring schemes and sequential tests. Next, we observe that the Type II errors are considerably reduced when more sensitive schemes are used (e.g. Type II error of the MaxSPRT-I-AA-N test is reduced from 0.13 to 0.02 when the *Linear* scheme is replaced by

Table 8.2: The quality metrics of the scoring scheme-test combinations that assume the uniform interleaving policy, measured on the dataset of interleaving experiments. $\triangle$ denotes a result that outperforms others in the same column (Wilcoxon test, $p < 0.01$). In bold are the best metric values in each column.

Scoring scheme	Test	Type I	Type II	$Acc_{B \succ A}$	$Acc_{A \succ B}$	$\mathbb{E}(T)$, days
Linear						
	OBF-I	**0.06**	0.04	0.92	0.97	2.85
	MaxSPRT-I-AA-N	**0.06**	0.13	0.88	0.81	2.43
Binary						
	OBF-I	0.08	0.03	0.95	0.97	2.70
	MaxSPRT-I-AA-N	**0.06**	0.09	0.91	0.86	2.07
Deduped						
	OBF-I	0.08	**0.00**	**0.97**	**0.98**	1.44
	MaxSPRT-I-AA-N	**0.06**	0.01	**0.97**	0.97	0.85
L_z						
	OBF-I	**0.06**	**0.00**	0.95	**0.98**	1.40
	MaxSPRT-I-AA-N	**0.06**	0.02	0.95	**0.98**	**0.83** $\triangle$

the L_z scoring scheme). Generally, we notice that the OBF-I test has lower Type II errors and higher accuracy than MaxSPRT-I-A-NN for all considered scoring schemes. However, this effect is achieved at the expense of longer deployment times. Indeed, for all the scoring schemes used, the MaxSPRT-based test stops earlier than OBF-I. For instance, when the *Linear* scoring scheme is used, MaxSPRT-I-AA-N stops, on average, after 2.43 days, while OBF-I stops after 2.85 days. Similarly, when *Deduped* scheme is used, MaxSPRT-I-AA-N stops after 0.85 days when OBF-I stops after 1.44 days.

From our earlier results in Table 6.5 (Section 6.11, page 115), it can be seen that the following ordering w.r.t. the scoring scheme sensitivity can be established: *Linear* $\prec$ *Binary* $\prec$ *Deduped* $\prec$ L_z. From Table 8.2 we observe the same ordering. Indeed, for any statistical test used, the experiments are deployed for the longest time when the *Linear* scoring scheme is used. Due to the higher sensitivity, experiments are stopped earlier if *Binary* is used. The fastest stopping times for both the OBF-I and MaxSPRT-I-AA-N tests are observed in the case of L_z, and these times are slightly smaller than the times observed for the *Deduped* scoring scheme. This is in agreement with Table 6.5 where the obtained sensitivity values for L_z and *Deduped* are close (mean z-scores are 2.33 and 2.36 for *Deduped* and L_z, respectively.)

Interestingly, the results reported in Table 8.2 for the combinations of the sequential tests with the *Binary* scheme are close to those reported in Chapter 7 (Table 7.1, page 136), where the same scheme is used. Indeed, the mean deployment times in the case of the OBF-I test reported are 2.70 (Table 8.2) and 2.66 (Table 7.1). The mean deployment times for the MaxSPRT-I-AA-N test are 2.07 (Table 8.2)

and 1.82 (Table 7.1). The observed differences are due to the variations in the experimental set-ups used in this chapter and in Chapter 7: (a) unlike Chapter 7, in this chapter interactions without clicks are not ignored, as this is incorrect in the case of interleaving with stratification, and (b) we use a slightly larger dataset in this chapter.

Overall, the results we obtained are in line with the experiments we report in our sensitivity optimisation study in Section 6.11 (Table 6.5, page 115) and our sequential testing study in Section 7.6 (Table 7.1, page 136). Indeed, we observed that MaxSPRT-I-AA-N exhibits a shorter deployment time, and more sensitive scoring schemes reduce the deployment times further. Moreover, the shortest mean deployment time is obtained when the most efficient MaxSPRT-I-AA-N test is used in combination with the most sensitive baseline scoring scheme, L_z, and allows to stop the interleaving experiments in our dataset after 0.83 days (20 hours), which corresponds to a 88% reduction in the deployment time in comparison with the standard week-long period. These observations allow us to answer RQ8.1 and RQ8.2. Indeed, with a sequential test fixed, increasing interleaving sensitivity reduces the mean deployment time. Similarly, with a scoring scheme fixed, MaxSPRT-I-AA-N achieves shorter deployment times than OBF-I.

8.5.2 Non-uniform Policy

Simulation In Table 8.3 we report the results we obtained in the evaluation experiments based on the simulation scenario (Section 8.4.2.2). From Table 8.3 we firstly notice that for all considered combinations of the Team Draft modifications and sequential tests, the Type I errors are well within the target upper bound of 0.05. Next, generally the Type II errors are higher than in previous experiments using the full experimental data (Table 8.2). In particular, the lowest level of 0.13 is achieved by the combination of L_z and F_z with the MaxSPRT-I-MC-N test and it is considerably higher than that in Table 8.3, where the combination of OBF-I and L_z has no Type II errors. This difference is caused by the decrease of the number of sessions.

Further, we observe that in Table 8.3 the accuracy metrics *AccBA* and *AccAB* have close values for all the considered combinations. In contrast, the expected deployment time metric $\mathbb{E}(T)$ changes considerably. Indeed, for each of the considered scoring schemes, the OBF-I test demonstrates markedly longer deployment times than MaxSPRT-I-MC-N, with differences of order 1.2 days (approximately 30 hours). On the other hand, for both OBF-I and MaxSPRT-I-MC-N using a more sensitive interleaving modification results in a lower deployment time. For instance, in the case of the OBF-I test, the deployment times for the L_z and F_z Team Draft modifications are equal to 3.75, and 3.61, respectively. The smallest value $\mathbb{E}(T)$ among the combinations with OBF-I is achieved in combination with F_z ($p < 0.01$). A

Table 8.3: The quality metrics of the considered scoring scheme/test combinations, measured on the dataset of interleaving experiments. These results are obtained by running the simulation-based evaluation scenario (Section 8.4). $\triangle$ denotes a result that outperforms others in the same column (Wilcoxon test, $p < 0.05$). In bold are the best metric values in each column.

Scoring scheme	Test	Type I	Type II	$Acc_{B \succ A}$	$Acc_{A \succ B}$	$\mathbb{E}(T)$, days
L_z						
	OBF-I	**0.00**	0.15	**0.88**	0.82	3.75
	MaxSPRT-I-MC-N	**0.00**	**0.13**	**0.88**	**0.87**	2.48
F_z						
	OBF-I	0.03	0.18	0.86	**0.87**	3.61
	MaxSPRT-I-MC-N	0.05	**0.13**	0.86	**0.87**	**2.37** $\triangle$

similar behaviour is observed for the MaxSPRT-I-MC-N test. The smallest value of $\mathbb{E}(T)$ is achieved by the combination of F_z and MaxSPRT-I-MC-N ($p < 0.05$, Wilcoxon test).

Overall, we observe the same behaviour as in Section 8.5.1: with a sequential test fixed, more sensitive scoring schemes achieve shorter mean deployment times; with a scoring scheme fixed, MaxSPRT-I-AA-N stops the experiments earlier than OBF-I. These two observations further answer RQ8.1 and RQ8.2.

Down-sampling In Table 8.4 we report the results for the experiments with the evaluation based on down-sampling, as discussed in Section 8.4.2.2. To reduce the effects of smaller datasets and noise due to sampling, we experiment in a scenario with daily stops, so that each test performs 7 interim stops. This allows a sufficient number of observations to be collected before the first stop after the down-sampling. Further, due to down-sampling, one of the long-term A/A experiments we use ceases to have reasonable amount of data, so we exclude it. To alleviate this, we use all the remaining 54 week-long A/A tests to train the stopping thresholds, and do not measure Type I errors.[53]

Again, from Table 8.4 we observe that the typical stopping times differ from the ones reported in Table 8.2, the main cause being that each experiment is down-sampled and has up to 16 times less data.

Further, we notice that for both scoring schemes L_z and F_z, the MaxSPRT-I-AA-N test stops experiments earlier than OBF-I and these differences are statistically significant for both scoring schemes ($p < 0.01$, Wilcoxon test). This is in agreement with the results obtained in Table 8.2. For instance, in the case of the L_z scoring, OBF-I terminates an experiment after 4.16 days on average, while MaxSPRT-I-AA-N stops after 3.45 days.

[53] An alternative approach would be to increase the number of cross-validation folds, so that we would have enough A/A data to adjust the stopping thresholds and to measure Type I errors. However, the down-sampling set-up is extremely computationally intensive and takes over a week in a highly parallel implementation. Hence, running it in e.g. 25-fold cross-validation loop would take several months. At the same time, across numerous experiments in Chapter 7 and in this section, we observed that Type I errors are always below or close to the required level α. Thus, we decided not to measure it in the down-sampling scenario.

Table 8.4: The quality metrics of the considered scoring scheme/test combinations, measured on the dataset of interleaving experiments. These results are obtained by running the down-sampling scenario (Section 8.4). In bold are the best metric values in each column. $\triangle$ denotes a result that outperforms others in the same column (Wilcoxon test, $p < 0.05$)

Scoring scheme	Test	Type II	$Acc_{B \succ A}$	$Acc_{A \succ B}$	$\mathbb{E}(T)$, days
L_z					
	OBF-I	0.18	0.78	0.83	4.16
	MaxSPRT-I-AA-N	**0.02**	0.79	0.80	3.46
F_z					
	OBF-I	0.14	**0.86**$\triangle$	**0.86**	3.91
	MaxSPRT-I-AA-N	**0.02**	0.78	0.80	**3.38**

At the same time, both MaxSPRT-I-AA-N and OBF-I stop the experiments earlier when F_z is used in comparison with the L_z scoring. In case of the OBF-I test, the deployment time is reduced from 4.16 days to 3.91 (6 hour reduction), and this difference is statistically significant ($p < 0.05$, Wilcoxon test). In the case of MaxSPRT-I-AA-N, the improvement is from 3.46 days to 3.38 (a reduction of approximately 2 hours). This observation is in line with our findings in Table 8.3, where we observed that F_z tends to stop earlier.

Overall, our observations from the experiments in the down-sampling scenario allow us to answer the stated research questions. With a sequential test fixed, using a more sensitive interleaving scheme (Generalised Team Draft F_z vs. L_z) results in a shorter deployment time (RQ8.1). Similarly, when the scoring scheme is fixed, by using MaxSPRT-I-AA-N instead of OBF-I we increase the interleaving efficiency (RQ8.2). Moreover, these observations hold in all evaluation experiments in this section.

Finally, now we are also able to answer RQ8.3. In the experiments in Section 8.5.1 (Table 8.2), we observed that the combination of L_z with MaxSPRT-I-AA-N achieves the shortest deployment time (0.83 days) among the combinations with the uniform policy. In our two following comparisons (Tables 8.3 and 8.4), we observed that the Generalised Team Draft combined with the MaxSPRT-I-based test demonstrates deployment times shorter than those of L_z combined with the same test. This answer the last research question RQ8.3: the combination of Generalised Team Draft and MaxSPRT-based test achieves the highest efficiency.

8.6 Conclusions

In this chapter our goal was to investigate if our previously proposed approaches to improve interleaving efficiency can be combined. The core of the sensitivity optimisation approach is the joint data-driven

optimisation of the interleaving scoring scheme and the interleaving policy. This approach is represented by our proposed Generalised Team Draft interleaving and was discussed in Chapter 6. In the second approach, we applied sequential tests that are able to stop the interleaving experiments early, once the observed data is sufficient to make a reliable outcome. This approach was discussed in Chapter 7.

In order to combine these two approaches, we started by adapting the sequential tests to the stratified outcome estimators used by Generalised Team Draft. In the next step, we investigated the efficiency of the four interleaving scoring schemes that assume the uniform interleaving policy in combination with the sequential testing procedures. Next, we selected the most sensitive scoring scheme, and compared it to Generalised Team Draft, varying the sequential test used. Since Generalised Team Draft generally uses a non-uniform policy, directly comparing it to the scoring schemes with the uniform interleaving policy is a hard task, as we only have a dataset of uniform-policy experiments. To perform this comparison, we followed two evaluation scenarios: simulation-based and down-sampling.

From our experiments we observed that the MaxSPRT-I-AA-N test, on average, stops the interleaving experiments earlier than the OBF-I test for all scoring schemes considered. By using MaxSPRT-I-AA-N in combination with the most sensitive scoring scheme with the uniform policy, proposed by Yue *et al.* (2010), the mean deployment time over our set of experiments can be reduced to 20 hours (Table 8.2). This corresponds to a 88% reduction in the deployment time in comparison with the standard 7-day scenario.

In further experiments we demonstrated that Generalised Team Draft achieves a shorter mean deployment time in combinations with both OBF-I and the MaxSPRT-based test than the machine-learned scoring scheme from (Yue *et al.*, 2010). Among the combinations of Generalised Team Draft and the machine-learned scoring scheme from (Yue *et al.*, 2010) with the OBF- and MaxSPRT-based tests, we observed that the combination of Generalised Team Draft and MaxSPRT-based tests achieved the shortest deployment time. This observation holds both in the simulation (Table 8.3, MaxSPRT-I-AA-N and Generalised Team Draft 2.37 days vs. MaxSPRT-I-AA-N and L_z 2.48 days, the second-based performance) and in the down-sampling (Table 8.4, MaxSPRT-I-AA-N and Generalised Team Draft 3.59 days vs. MaxSPRT-I-AA-N and L_z 3.71 days, the second-best performance) scenarios.

Overall, our study allowed to answer the question stated in Section 8.1. Indeed, we observed that the sensitivity optimisation and the sequential testing approaches can be effectively combined and produce a cumulative improvement in the evaluation efficiency. In this chapter we demonstrated that by relying on the historical interaction data recorded in online experiments, we can simultaneously optimise the interleaving parameters using the Generalised Team Draft framework (Chapter 6), adjust the stopping thresholds for the sequential tests (Chapter 7) and achieve considerable improvements in the online

evaluation efficiency. Hence, these results support the statement of this thesis (Section 1.3). Our work in this chapter closes the last point of the roadmap for improving the efficiency of the evaluation pipeline that we outlined in Section 3.5. In the next chapter, we conclude this thesis.

Chapter 9

Conclusions and Future Work

With the growth in the scale of operation of the modern commercial search engines and the rise of the data-centric evaluation culture (Tang *et al.*, 2010), requirements for search engine evaluation pipelines evolved considerably over the past years. In particular, efficiency and scalability are becoming increasingly important (Kohavi *et al.*, 2009; Tang *et al.*, 2010), with the number of performed online evaluation experiments quickly rising over time (Kohavi *et al.*, 2013).

In this thesis, we studied how the evaluation pipeline for a web search engine can be improved by re-using historical interaction data that is routinely collected by search engines. In particular, we split a typical evaluation pipeline into three consecutive steps: offline evaluation, online experiment scheduling, and online evaluation. After that we discussed how each of these steps can be improved.

Our goal in improving the offline evaluation step (Chapter 4) was to develop offline evaluation methods that are highly aligned with online user satisfaction indicators. Such an alignment would result in less experiments being deployed and rejected by the later steps of the evaluation pipeline. In order to achieve this, we proposed offline user model-based evaluation metrics for query auto-completion mechanisms. We proposed to train the underlying models against datasets of historical interaction data, thus allowing the models to accurately reflect the online user behaviour.

In Chapter 5, we discussed how the online scheduling step can be improved, so that more successful experiments are deployed in a unit of time. Clearly, this goal has a direct connection to the efficiency of the evaluation pipeline. Indeed, the optimised scheduling prioritises the promising experiments, so that the limited resource of user interactions is spent strategically.

Further, in Chapters 6, 7, and 8, we concentrated on improving the efficiency of the online evaluation step. We approached this task from two distinct perspectives. Firstly, we worked on increasing the sensitivity of online evaluation experiments and extending their applicability to new domains. Specifically,

we proposed the Generalised Team Draft interleaving framework (Chapter 6) that increases the sensitivity of interleaving experiments by a joint optimisation of the interleaving parameters. Generalised Team Draft also extends the applicability of the sensitive interleaving methods to domains with grid-based representation, e.g. image search. Secondly, we discussed how sequential statistical tests (Chapter 7) can be used to increase the efficiency of the online evaluation by stopping online experiments when the collected data is sufficient to make a reliable conclusion.

Throughout this work, we thoroughly evaluated each of the improved steps. Moreover, we empirically demonstrated that the improvements of the last two steps (interleaving sensitivity and sequential testing) can be combined together (Chapter 8). As a result, even higher gains in the interleaving efficiency can be achieved.

In the remainder of this chapter we review the contributions of this thesis in Section 9.1 and discuss its conclusions in Section 9.2. We conclude this chapter in Section 9.3 with a discussion of possible directions of future work that can stem from this thesis.

9.1 Summary of Contributions

The main contributions of this thesis are the following:

A family of offline query auto-completion evaluation metrics In Chapter 4, we concentrated on improving the offline evaluation step for query auto-completion (QAC) mechanisms. We proposed a family of offline metrics for the evaluation of the query auto-completion mechanisms, *Saved*, and two metrics of this family, *pSaved* and *eSaved*. The *pSaved* metric predicts the probability of the user using the query auto-completions mechanism, and *eSaved* reflects the expected ratio of the keypresses the QAC mechanism saves the user from typing. We experimented with instantiations of these metrics that are based on the model of the user interactions proposed in Section 4.3. This model is trained on historical interaction data and aims to reflect the patterns of the user's QAC examination behaviour.

Optimised scheduling of online experiments In Chapter 5, we stated the problem of the optimal scheduling of the online experiments. 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 experimented with a diverse set of features that can be used to train the learning-to-rank model using historical experimentation data, and evaluated several scheduling strategies based on simple features, e.g. effectiveness metrics and exploratory deployment outcomes.

Sensitive and general interleaving framework In Chapter 6, we addressed an important problem of improving the interleaving sensitivity. We started with introducing a user model-based approach for increasing the interleaving sensitivity on a per-query level (Section 6.3). After discussing the limitations of this initial approach, we proposed the Generalised Team Draft interleaving framework that overcomes these limitations and generalises the existing research in two aspects (Section 6.4). First, it achieves an increased sensitivity by performing a joint optimisation of the click credit assignment function and the interleaving policy on the historical interaction data. Second, it is formulated to be general with respect to the manner the results are presented, thus it can be applied in domains with a grid-based representation, such as image search. A part of this generalisation is a new unbiasedness criterion, that can be applied for grid-based domains and machine-learned click scoring schemes. Finally, we proposed to use a stratified estimate of the experiment outcome, as it both simplifies the optimisation problem and increases sensitivity in some cases (Section 6.6).

Sequential tests for online evaluation In Chapter 7, we continued to work on the efficiency of the online experiments, as it is the most time-consuming step of the evaluation pipeline (Chapter 3). In particular, we studied how sequential testing procedures can be adapted to reduce the time online evaluation experiments require. These procedures are designed so that they can stop online experiments when the observed data is sufficient to make a reliable conclusion about the experiment's outcome. In Section 7.3 we proposed a modification of the O'Brien & Fleming group sequential test (OBF) that can be applied to an interleaving evaluation. Further, we described the MaxSPRT-based tests that adjust their stopping thresholds w.r.t. to a dataset of historical interaction data obtained from A/A experiments.

Combination of the sensitivity optimisation and sequential testing methods In Chapter 8, we investigated how the approaches to improve the interleaving efficiency we discussed earlier (sensitivity optimisation, Chapter 6, and sequential testing, Chapter 7) can be combined. In order to combine these two approaches, we started by adapting the sequential tests to the stratified outcome estimators used by Generalised Team Draft. In the next step, we investigated the efficiency of the interleaving scoring schemes that assume the uniform interleaving policy in combination with our proposed sequential testing procedures. Next, we selected the most sensitive scoring scheme, and compared it to Generalised Team Draft, by varying the sequential test used.

9.2 Summary of Conclusions

In this section, we summarise the conclusions of the individual chapters of this thesis. These conclusions support the statement of the thesis as formulated in Section 1.3 and demonstrate that the historical

interaction data can be re-used to improve each of the individual steps of the evaluation pipeline.

Offline evaluation metrics for QAC In Chapter 4.9.2 we compared our proposed *pSaved* and *eSaved* metrics for the query auto-completion evaluation to the existing query auto-completion metrics. We experimented in two evaluation scenarios. We evaluated the weighted correlation of the proposed metrics with the online query auto-completion click-through rate. In the second scenario, we used historical interaction data to simulate A/B tests.

Our experiments in Section 4.9 demonstrated that our proposed metrics achieve significantly higher correlation with the online success rate metric than the existing offline metrics. In particular, the *pSaved* metric instantiated with the prefix-independent model of user examination behaviour (f_l^i, Section 4.3) achieved the weighted correlation level of 0.904 with the success rate indicator (Table 4.4). In the series of A/B test simulation experiments reported in Table 4.5, the differences in the *pSaved* metric combined with the prefix length-dependent examination function f_l^d achieved the highest correlation with the differences in the success rate (0.820).

Optimised scheduling of online experiments Our findings in Section 5.7 suggest that our proposed machine-learned schedule optimisation algorithms outperform the "natural" (randomised) schedule when the number of the successful experiments performed under a limited number of available user interactions is measured. More specifically, we demonstrated that an experiment scheduling algorithm based on the gradient boosted regression trees (PGBM) that combines various features achieved the highest scheduling quality. Indeed, PGBM achieves an AUC of 0.86 (Table 5.2) in discriminating successful interleaving experiments from unsuccessful ones. PGBM outperforms other algorithms in the task of deploying the maximum number of successful experiments under the contained number of interactions (Table 5.3).

Our study also showed that a simple scheduler that ranks experiments according to their ranking effectiveness can achieve a smaller, but still a marked improvement over the "natural" random baseline. Interestingly, we observed that the scheduling algorithm that ranks interleaving experiments according to the outcomes of the preliminary short deployments can improve the scheduling efficiency dramatically in comparison with the natural stochastic ordering of the experiment queue (e.g. 80% improvement when 5% of the experimental budged is used for the exploration, left part of Table 5.3).

Generalised Team Draft framework In our evaluation study in Section 6.11, we compared our proposed Generalised Team Draft interleaving framework to the existing state-of-the-art baselines. This study was performed in both the document and image search domains. In this study, we demonstrated

that our proposed framework achieves the highest sensitivity on both datasets, outperforming the heuristic (Chapelle *et al.*, 2012) and machine-learned (Yue *et al.*, 2010) baselines.

In particular, our experiments demonstrated that on the document search interleaving dataset, Generalised Team Draft achieves the median relative z-score of 2.15 while the best performing baseline proposed by Yue *et al.* (2010) has the median relative z-score of 2.09 (Table 6.5). Similarly, Generalised Team Draft proved to be more sensitive in our experiments on the image search dataset (Table 6.8, median relative z-score 1.18). Furthermore, the stratified outcome estimator that we proposed allowed us to increase the sensitivity of the baseline scoring schemes in some cases. For instance, the *Binary* scoring scheme increased its median relative z-score from 0.98 to 1.04 (Table 6.5). However, in the case of the *Deduped* baseline scoring scheme the stratification did not increase sensitivity.

Sequential testing for online experimentation In our evaluation study we experimented with sequential testing methods in the context of interleaving experimentation (Section 7.6.1).

Our study demonstrated that by using our proposed sequential tests, a marked reduction in the duration of the interleaving experiments can be achieved, without significant losses in other metrics, such as the Type I and Type II error probabilities. Further, we observed that the MaxSPRT-based tests demonstrated the shortest mean deployment times, in comparison to the OBF-based tests. From our results it follows that by training the stopping thresholds on a dataset of A/A comparisons we can improve the performance of the MaxSPRT-based tests.

Specifically, from our experimental study in Section 7.6, we observed that the MaxSPRT-based tests that adjust their stopping thresholds using A/A experiments, MaxSPRT-I-AA and MaxSPRT-I-AA-N, achieve the shortest mean deployment time (1.82 days, 74% improvement over the typical 7-day scenario, Table 7.1) without significantly degrading other metrics.

Combining sensitivity optimisation and sequential testing Our study in Chapter 8 demonstrated that by combining the interleaving sensitivity optimisation (Chapter 6) and sequential testing (Chapter 7), one can achieve cumulative gains in the interleaving efficiency.

From our experimental results in Section 8.5, we firstly observed that by using increasingly sensitive interleaving scoring schemes with a sequential test fixed, one can achieve progressively higher efficiency. Similarly, for each of the considered scoring schemes, the MaxSPRT-based tests demonstrated shorter mean deployment times than the OBF-I test. In particular, among various combinations with the interleaving scoring schemes that use the uniform interleaving policy, the combination of the MaxSPRT-I-AA-N test and the machine-learned scoring scheme L_z (Yue *et al.*, 2010) achieves the high-

est efficiency. This combination, on average, stops the experiments after 20 hours (Table 8.2), which corresponds to a 88% reduction in the deployment time in comparison with the standard 7-day scenario.

To further compare L_z to Generalised Team Draft, we used two evaluation scenarios, which are required as Generalised Team Draft uses a non-uniform interleaving policy, while only a dataset with the uniform-policy experiments is available to us. In both scenarios, Generalised Team Draft outperforms L_z when both combinations with the OBF-I and MaxSPRT-based tests are considered (Table 8.3 and 8.4). For instance, in the down-sampling scenario reported in Table 8.4, the combination of Generalised Team Draft with OBF-I achieves the mean deployment time of 3.91 days, while the combination of L_z and OBF-I stops the experiments after 4.16 days.

Summary Overall, our experiments support the statement of this thesis. Indeed, by using historical interaction data, (a) we increased the agreement between offline and online metrics, so that less unsuccessful experiments would proceed to the later evaluation stages, (b) we optimised the scheduling of the online experiments, so that under limited budget only the likely successful online experiments would be deployed, (c) we improved the sensitivity of the interleaving experiments, so that each experiment can be deployed for a shorter time, and (d) we optimised stopping thresholds of the sequential testing procedures, so that interleaving experiment are stopped earlier, on average. Finally, we have demonstrated that improvements in interleaving sensitivity and advanced sequential testing add up so that the efficiency of the pipeline as a whole is increased. Each of these improvements results in an increased efficiency of the evaluation pipeline as a whole, allowing search engines to progress at a higher rate.

9.3 Directions for Future Work

The web search evaluation as an area of research has both a rich history and a bright future. Our work in this thesis is merely an attempt to address some of the existing gaps. Further in this section, we discuss some of the future research directions that remain open and can build upon our work.

Interleaving and A/B testing In this thesis, we adopted a very practical, data-driven approach to improving interleaving, primarily by optimising the scoring scheme and interleaving policy. A major downside of this approach is that it does not allow us to get in-depth insights about the inner structure of interleaving as a method, its trade-offs and limitations. For instance, it is clear that by inserting results obtained from an experimental ranker we might degrade the users' search experience. How can we reduce this risk, and how could that affect the interleaving sensitivity? How can we control the trade-off between the sensitivity and this risk, and how one can define an optimum point?

Moreover, what are the ground-truth decisions newly developed interleaving methods should agree with? It is known that click-based interleaving scores can be biased (Hofmann, Behr & Radlinski, 2012) and some effort has been applied to make the interleaving outcomes consistent with interaction-level A/B test metrics (Schuth *et al.*, 2015). However, the interaction-level metrics can contradict each other, they are not always interpretable, and it is not clear if they can be considered as the ultimate ground-truth. Hence, a better ground-truth is needed.

We believe that one of the possible approaches to understand interleaving better is to re-design interleaving from the beginning, based on an axiomatic approach. Assuming that the users behave according to a particular click model, that we fix a particular level of the risk we are willing to accept, and specify the ground-truth metric — what interleaving method would we come up with? Radlinski & Craswell (2013) made an important step in that direction and it was further developed in our Generalised Team Draft framework (Chapter 6), but we believe there is much more to do. The central idea of this thesis — re-using the historical interaction data to improve the evaluation methods — can play an important role in evaluating such an interleaving method and optimising its parameters.

Another important direction of research, that can build on top of our work and that of Yue *et al.* (2010), is the sensitive machine-learned online metrics. So far we only scratched the surface of this direction in Chapter 6. Indeed, the nature of the user's interaction with the search engine is intrinsically sequential and multi-faceted. To build a powerful online evaluation metric, one would need to model not only the user's clicks, but also the mouse movements, interactions with query auto-completions, queries, etc. This model might analyse the user's behaviour spanning over numerous sessions and even the whole history of the user's interactions with the search engine. However, how can one guarantee the unbiasedness of such a metric, in interleaving and A/B tests? One the other hand, if one is able to build such a rich model of the user's online behaviour, in a plausible scenario it can also be used to simulate users offline and form a foundation to an offline quality metric. These questions might be approached by using sequence learners such as recurrent neural networks in the spirit of Borisov *et al.* (2016), but this direction clearly needs a further in-depth investigation.

Pipeline optimisation In this thesis, we optimised the steps of the evaluation pipeline independently, optimising each step while assuming that others are fixed. However, the steps are inter-dependent. To illustrate this, recall that the online evaluation step is performed for the ranking changes that (a) are optimised against an offline evaluation metric, and (b) do not decrease the ranking quality in an offline evaluation. Clearly, this filtering determines the space of possible ranking changes that are evaluated in the later stages, in particular during the online evaluation step. For instance, if the offline evaluation

metric or the learning-to-rank target starts to favour attractive, but possibly not relevant results, this would affect the optimal online evaluation metric, as it would need to account for this change.

Ideally, all the steps of the pipeline need to be optimised simultaneously, in a single optimisation process with the aim to increase efficiency and reduce the number of errors. In some sense, this thesis discusses a "coordinate-wise descent" in the space of "parameters" of the evaluation pipeline, which is not necessary the ideal approach. However, such a task appears to be tremendously involved, particularly from the data collection perspective and represents a significant challenge.

Overall, a significant number of important research questions remain open in the area of web search evaluation and we believe they should be tackled in future studies, possibly relying on our work in this thesis as a foundation.

Bibliography

Alonso, O., Rose, D. E. & Stewart, B. (2008). Crowdsourcing for relevance evaluation. *In* 'SIGIR Forum'. Vol. 42. 2.2

Arguello, J., Diaz, F., Callan, J. & Crespo, J.-F. (2009). Sources of evidence for vertical selection. *In* 'Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval'. 15

Arkhipova, O., Grauer, L., Kuralenok, I. & Serdyukov, P. (2015). Search engine evaluation based on search engine switching prediction. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.1

Asmussen, S. & Glynn, P. W. (2007). *Stochastic simulation: Algorithms and analysis*. Vol. 57. Springer Science & Business Media. 6.2.1

Bar-Yossef, Z. & Kraus, N. (2011). Context-sensitive query auto-completion. *In* 'Proceedings of the 20th International Conference on World Wide Web'. 1.1, 21, 4.2.4, 4.5, 4.7.3

Barreda-Ángeles, M., Arapakis, I., Bai, X., Cambazoglu, B. B. & Pereda-Baños, A. (2015). Unconscious physiological effects of search latency on users and their click behaviour. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.2

Bartroff, J., Lai, T. L. & Shih, M.-C. (2012). *Sequential Experimentation in Clinical Trials: Design and Analysis*. Vol. 298. Springer. 7.2

Bennett, P. N., Radlinski, F., White, R. W. & Yilmaz, E. (2011). Inferring and using location metadata to personalize web search. *In* 'Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.1

Borisov, A., Markov, I., de Rijke, M. & Serdyukov, P. (2016). A distributed representation approach to modeling user browsing behavior in web search. *In* 'Proceedings of the 25th International Conference on World Wide Web'. 9.3

Burges, C. J. (2010). From RankNet to LambdaRank to LambdaMART: An overview. *MSR Technical Rep. MSR-TR-2010-82*. 3.2

Carterette, B. & Jones, R. (2008). Evaluating search engines by modeling the relationship between relevance and clicks. *In* 'Advances in Neural Information Processing Systems'. pp. 217–224. 2.2.1

Chakraborty, S., Radlinski, F., Shokouhi, M. & Baecke, P. (2014). On correlation of absence time and search effectiveness. *In* 'Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.1

Chapelle, O. & Zhang, Y. (2009). A dynamic bayesian network click model for web search ranking. *In* 'Proceedings of the 18th International Conference on World Wide Web'. 2.2.1, 2.5, 2.5, 2.5, 2.3, 4.2.1, 5.4.2, 5.5, 6.3.2

Chapelle, O., Ji, S., Liao, C., Velipasaoglu, E., Lai, L. & Wu, S.-L. (2011). Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval*. 2.2.1, 4.2.3

Chapelle, O., Joachims, T., Radlinski, F. & Yue, Y. (2012). Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems*. 1.1, 2.3, 2.3.2, 2.3.2, 2.3.3, 2.4, 5.2, 5.8, 6.1, 6.2.1, 6.3.3, 6.6, 6.6, 6.7, 6.8, 6.10.1, 6.10.2, 6.10.2, 6.11.1, 7.3, 9.2

Chapelle, O., Metlzer, D., Zhang, Y. & Grinspan, P. (2009). Expected reciprocal rank for graded relevance. *In* 'Proceedings of the 18th ACM International Conference on Information and Knowledge Management'. 2.2, 2.2.1, 2.4, 2.5, 3.2, 4.1, 4.2.1, 4.2.3, 4.6, 25, 4.6, 4.7.2, 4.7.2, 5.4.2

Chuklin, A., Markov, I. & de Rijke, M. (2015). *Click Models for Web Search*. Morgan & Claypool. 2.5, 4.7.1, 6.3.2

Chuklin, A., Schuth, A., Hofmann, K., Serdyukov, P. & de Rijke, M. (2013). Evaluating aggregated search using interleaving. *In* 'Proceedings of the 22nd ACM International Conference on Information and Knowledge Management'. 4.7.2, 6.2.2

Chuklin, A., Serdyukov, P. & de Rijke, M. (2013). Click model-based information retrieval metrics. *In* 'Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.2.1, 2.4, 4.2.1, 4.2.3, 4.6, 25, 4.7.2

Clarke, C. L., Craswell, N. & Soboroff, I. (2009). Overview of the TREC 2009 Web Track. *In* 'Proceedings of the 18th Text REtrieval Conference'. 2.2

Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S. & MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.3

Cleverdon, C. (1967). The cranfield tests on index language devices. *Aslib Proceedings* **19**(6), 173–194. 1.1, 2.2, 2.6

Cleverdon, C. W. & Keen, M. (1966). Aslib cranfield research project-factors determining the performance of indexing systems; volume 2, test results. (document), 2.1, 2.2.1

Collins-Thompson, K., Bennett, P., Diaz, F., Clarke, C. & Voorhees, E. M. (2013). TREC 2013 Web Track overview. *In* 'Proceedings of the 22nd Text REtrieval Conference'. 2.2, 2.2.1

Collins-Thompson, K., Macdonald, C., Bennett, P., Diaz, F. & Voorhees, E. M. (2014). TREC 2014 Web Track overview. *In* 'Proceedings of the 23rd Text REtrieval Conference'. 2.2, 2.2.1

Cormack, G. V., Palmer, C. R. & Clarke, C. L. (1998). Efficient construction of large test collections. *In* 'Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.2

Cramér, H. (1946). *Mathematical methods of statistics*. Princeton University Press. 13

Craswell, N., Zoeter, O., Taylor, M. & Ramsey, B. (2008). An experimental comparison of click position-bias models. *In* 'Proceedings of the 1st International Conference on Web Search and Data Mining'. 2.2.1, 2.5, 4.3, 4.3, 4.4, 6.3.2, 43

Croft, W. B., Metzler, D. & Strohman, T. (2010). *Search engines: Information retrieval in practice*. Addison-Wesley Publishing Company. 2.2.1, 2.2.1

Deng, A., Xu, Y., Kohavi, R. & Walker, T. (2013). Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. *In* 'Proceedings of the 6th ACM International Conference on Web Search and Data Mining'. 6.2.1

Diaz, F. (2009). Integration of news content into web results. *In* 'Proceedings of the Second ACM International Conference on Web Search and Data Mining'. 15

Drutsa, A., Ufliand, A. & Gusev, G. (2015). Practical aspects of sensitivity in online experimentation with user engagement metrics. *In* 'Proceedings of the 24th ACM International on Conference on Information and Knowledge Management'. 1.1, 2.3.1, 2.4, 5.8, 6.1

Duan, H. & Hsu, B.-J. P. (2011). Online spelling correction for query completion. *In* 'Proceedings of the 20th International Conference on World Wide Web'. 4.2.4, 4.7.3

Dupret, G. & Lalmas, M. (2013). Absence time and user engagement: evaluating ranking functions. *In* 'Proceedings of the 6th ACM International Conference on Web Search and Data Mining'. 2.3.1

Dupret, G. E. & Piwowarski, B. (2008). A user browsing model to predict search engine click data from past observations.. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.5, 4.7.1, 6.3.2

Dupret, G., Murdock, V. & Piwowarski, B. (2007). Web search engine evaluation using clickthrough data and a user model. *In* 'WWW2007 workshop Query Log Analysis: Social and Technological Challenges'. 2.5

Grotov, A., Whiteson, S. & de Rijke, M. (2015). Bayesian ranker comparison based on historical user interactions. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 18

Guo, F., Liu, C., Kannan, A., Minka, T., Taylor, M., Wang, Y.-M. & Faloutsos, C. (2009). Click chain model in web search. *In* 'Proceedings of the 18th International Conference on World Wide Web'. 2.5, 6.3.2, 35

Harman, D. (1994). Overview of the second Text REtrieval Conference. *In* 'Proceedings of the Workshop on Human Language Technology'. 2.2.1

Hassan, A., Jones, R. & Klinkner, K. L. (2010). Beyond dcg: user behavior as a predictor of a successful search. *In* 'Proceedings of the 3rd ACM International Conference on Web Search and Data Mining'. 2.3.1

Hofmann, K., Behr, F. & Radlinski, F. (2012). On caption bias in interleaving experiments. *In* 'Proceedings of the 21st ACM International Conference on Information and Knowledge Management'. 9.3

Hofmann, K., Mitra, B., Radlinski, F. & Shokouhi, M. (2014). An eye-tracking study of user interactions with query auto completion. *In* 'Proceedings of the 23rd ACM International Conference on Information and Knowledge Management'. 4.2.2

Hofmann, K., Whiteson, S. & de Rijke, M. (2011). A probabilistic method for inferring preferences from clicks. *In* 'Proceedings of the 20th ACM International Conference on Information and Knowledge Management'. 2.3.2, 2.3.2, 6.2

Hofmann, K., Whiteson, S. & de Rijke, M. (2012). Estimating interleaved comparison outcomes from historical click data. *In* 'Proceedings of the 21st ACM International Conference on Information and Knowledge Management'. 5.2

Järvelin, K. & Kekäläinen, J. (2000). IR evaluation methods for retrieving highly relevant documents. *In* 'Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.2.1

Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*. 2.2, 2.2.1, 3.2, 24, 5.4.2

Joachims, T. (2002). Optimizing search engines using clickthrough data. *In* 'Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 2.1, 2.3.2, 2.5, 5.4.3, 6.2

Joachims, T. (2003). Evaluating retrieval performance using clickthrough data. *In* 'Text Mining'. Physica/Springer Verlag. 1.1, 2.5, 6.2, 6.6

Joachims, T., Granka, L., Pan, B., Hembrooke, H. & Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. *In* 'Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.5

Johari, R., Pekelis, L. & Walsh, D. J. (2015). Always valid inference: Bringing sequential analysis to A/B testing. *arXiv preprint arXiv:1512.04922*. 7.2

Jones, E., Oliphant, T. & Peterson, P. (2016). SciPy: Open source scientific tools for Python, 2001–. *URL http://www. scipy. org*. 6.7

Kantor, P. B. & Voorhees, E. M. (1996). Report on the TREC-5 Confusion Track. *In* 'Proceedings of the 5th Text REtrieval Conference'. 2.2.1

Kent, A., Berry, M. M., Luehrs, F. U. & Perry, J. W. (1955). Machine literature searching viii. operational criteria for designing information retrieval systems. *American documentation* **6**(2), 93–101. (document), 2.1, 2.2.1

Kharitonov, E. (2014). Improving offline and online web search evaluation by modelling the user behaviour. *In* 'Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.5

Kharitonov, E. & Serdyukov, P. (2012). Demographic context in web search re-ranking. *In* 'Proceedings of the 21st ACM International Conference on Information and Knowledge Management'. 1.1

Kharitonov, E., Macdonald, C., Serdyukov, P. & Ounis, I. (2013*a*). Intent models for contextualising and diversifying query suggestions. *In* 'Proceedings of the 22nd ACM International Conference on Information and Knowledge Management'. 21, 4.5

Kharitonov, E., Macdonald, C., Serdyukov, P. & Ounis, I. (2013*b*). User model-based metrics for offline query suggestion evaluation. *In* 'Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.5, 4.1, 4.2.2

Kharitonov, E., Macdonald, C., Serdyukov, P. & Ounis, I. (2013*c*). Using historical click data to increase interleaving sensitivity. *In* 'Proceedings of the 22nd ACM International Conference on Information and Knowledge Management'. 1.5, 6.1, 44

Kharitonov, E., Macdonald, C., Serdyukov, P. & Ounis, I. (2015*a*). Generalized Team Draft interleaving. *In* 'Proceedings of the 24th ACM International Conference on Information and Knowledge Management'. 1.5, 6.1

Kharitonov, E., Macdonald, C., Serdyukov, P. & Ounis, I. (2015*b*). Optimised scheduling of online experiments. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.5, 5.1

Kharitonov, E., Vorobyev, A., Macdonald, C., Serdyukov, P. & Ounis, I. (2015). Sequential testing for early stopping of online experiments. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.5, 7.1

Kohavi, R. (2012). Online controlled experiments: introduction, learnings, and humbling statistics. *In* 'Proceedings of the 6th ACM Conference on Recommender systems'. 3.4

Kohavi, R., Crook, T., Longbotham, R., Frasca, B., Henne, R., Ferres, J. L. & Melamed, T. (2009). Online experimentation at microsoft. *Data Mining Case Studies* p. 11. 1.1, 3.1, 9

Kohavi, R., Deng, A., Frasca, B., Longbotham, R., Walker, T. & Xu, Y. (2012). Trustworthy online controlled experiments: Five puzzling outcomes explained. *In* 'Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 1.1, 2.3.1, 6.1, 7.4

Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y. & Pohlmann, N. (2013). Online controlled experiments at large scale. *In* 'Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 1.1, 1.2, 2.4, 5.1, 5.3, 5.3, 5.8, 6.1, 7.2, 9

Kohavi, R., Deng, A., Longbotham, R. & Xu, Y. (2014). Seven rules of thumb for web site experimenters. *In* 'Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 5

Kulldorff, M., Davis, R. L., Kolczak, M., Lewis, E., Lieu, T. & Platt, R. (2011). A maximized sequential probability ratio test for drug and vaccine safety surveillance. *Sequential Analysis* **30**(1), 58–78. 7.1, 7.2, 7.3.2, 7.3.2, 7.3.2, 8.1, 8.4

Lan, K. G. & DeMets, D. L. (1983). Discrete sequential boundaries for clinical trials. *Biometrika* **70**(3), 659–663. 7.2

Leung, J. Y. (2004). *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press. 5.3

Li, L., Kim, J. Y. & Zitouni, I. (2015). Toward predicting the outcome of an A/B experiment for search relevance. *In* 'Proceedings of the 8th ACM International Conference on Web Search and Data Mining'. 18, 5.2

Li, Y., Dong, A., Wang, H., Deng, H., Chang, Y. & Zhai, C. (2014). A two-dimensional click model for query auto-completion. *In* 'Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.2, 4.2.4

Ling, C. X., Huang, J. & Zhang, H. (2003). Auc: a statistically consistent and more discriminating measure than accuracy. *In* 'The 12th International Joint Conference on Artificial Intelligence'. 5.6.1

Loptev, A., Selugina, A. & Starikovskaya, T. (2014). Simple and efficient string algorithms for query suggestion metrics computation. *In* 'Proceeding of the 21st International Symposium on String Processing and Information Retrieval'. 4.6

Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. 2.2

Markov, I., Kharitonov, E., Nikulin, V., Serdyukov, P., de Rijke, M. & Crestani, F. (2014). Vertical-aware click model-based effectiveness metrics. *In* 'Proceedings of the 23rd ACM International Conference on Information and Knowledge Management'. 4.7.2

Megorskaya, O., Kukushkin, V. & Serdyukov, P. (2015). On the relation between assessor's agreement and accuracy in gamified relevance assessment. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.2

Mitra, B. (2015). Exploring session context using distributed representations of queries and reformulations. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.4

Mitra, B., Shokouhi, M., Radlinski, F. & Hofmann, K. (2014). On user interactions with query autocompletion. *In* 'Proceedings of the 37th international ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.2

Moffat, A. & Zobel, J. (2008). Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*. 2.2.1, 2.5

Moffat, A., Webber, W. & Zobel, J. (2007). Strategic system comparisons via targeted relevance judgments. *In* 'Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.2

O'Brien, P. C. & Fleming, T. R. (1979). A multiple testing procedure for clinical trials. *Biometrics*. 7.1, 7.2, 7.3.1, 7.3.1, 8.1, 8.4

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830. 5.4.3

Pocock, S. J. (1977). Group sequential methods in the design and analysis of clinical trials. *Biometrika* **64**(2), 191–199. 7.2

Radlinski, F. & Craswell, N. (2010). Comparing the sensitivity of information retrieval metrics. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.2, 4.2.3, 5.2, 6.1, 7.3

Radlinski, F. & Craswell, N. (2013). Optimized interleaving for online retrieval evaluation. *In* 'Proceedings of the 6th ACM International Conference on Web Search and Data Mining'. 2.3.2, 2.3.2, 2.3.2, 6.1, 6.2, 6.2.2, 6.3, 6.3.1, 6.3.1, 6.3.1, 6.4, 6.4, 6.4, 6.5, 6.6, 6.10.1, 9.3

Radlinski, F., Kurup, M. & Joachims, T. (2008). How does clickthrough data reflect retrieval quality?. *In* 'Proceedings of the 17th ACM Conference on Information and Knowledge management'. 2.3.1, 2.3.2, 2.3.2, 2.2, 6.2, 6.4, 6.6

Ridgeway, G. (2004). The gbm package. *R Foundation for Statistical Computing, Vienna, Austria*. 5.4.3

Robert, C. P. & Casella, G. (2009). *Introducing Monte Carlo Methods with R (Use R)*. 1st edn. Springer-Verlag. Berlin, Heidelberg. 6.2.1

Sanderson, M. (2010). *Test collection based evaluation of information retrieval systems*. Now Publishers. (document), 2.2, 2.1, 2.2.1, 2.2.1, 2.2.1

Sanderson, M., Paramita, M. L., Clough, P. & Kanoulas, E. (2010). Do user preferences and evaluation measures line up?. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.3

Schuth, A., Hofmann, K. & Radlinski, F. (2015). Predicting search satisfaction metrics with interleaved comparisons. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3, 2.3.3, 6.1, 6.8, 9.3

Shokouhi, M. & Radinsky, K. (2012). Time-sensitive query auto-completion. *In* 'Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 4.2.4, 4.5, 4.7.3

Shokouhi, M., White, R. W., Bennett, P. & Radlinski, F. (2013). Fighting search engine amnesia: Reranking repeated results. *In* 'Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.1

Siegmund, D. (1985). *Sequential analysis: tests and confidence intervals*. Springer. 7.2

Sparck Jones, K. & van Rijsbergen, C. J. (1975). Report on the need for and provision of an "ideal" Information Retrieval test collection. Technical Report 5266. *Computer Laboratory, University of Cambridge*. 2.2

Strizhevskaya, A., Baytin, A., Galinskaya, I. & Serdyukov, P. (2012). Actualization of query suggestions using query logs. *In* 'Proceedings of the 21st International Conference on World Wide Web'. 4.2.4, 4.5

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press. 5.4.2

Tang, D., Agarwal, A., O'Brien, D. & Meyer, M. (2010). Overlapping experiment infrastructure: More, better, faster experimentation. *In* 'Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 1.2, 6, 3.4, 5.2, 9

Thomas, P. & Hawking, D. (2006). Evaluation by comparing result sets in context. *In* 'Proceedings of the 15th ACM International Conference on Information and Knowledge Management'. 3.2

Tran-Thanh, L., Chapman, A., Munoz De Cote Flores Luna, J. E., Rogers, A. & Jennings, N. R. (2010). Epsilon–first policies for budget–limited multi-armed bandits. *In* 'The 24th AAAI Conference on Artificial Intelligence'. 5.4.2

Van Rijsbergen, C. J. (1974). Foundation of evaluation. *Journal of Documentation* **30**(4), 365–373. 2.2.1

Voorhees, E. (2002). The philosophy of information retrieval evaluation. *In* C. Peters, M. Braschler, J. Gonzalo & M. Kluck, eds, 'Evaluation of Cross-Language Information Retrieval Systems'. Vol. 2406 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. pp. 355–370. 1.1, 2.2, 2.6

Wald, A. (1945). Sequential tests of statistical hypotheses. *Ann. Math. Statist.* **16**(2), 117–186. 7.2, 7.3.2

Wald, A. & Wolfowitz, J. (1948). Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics* pp. 326–339. 7.2

Wang, S. K. & Tsiatis, A. A. (1987). Approximately optimal one-parameter boundaries for group sequential trials. *Biometrics*. 7.2

Yilmaz, E., Shokouhi, M., Craswell, N. & Robertson, S. (2010). Expected browsing utility for web search evaluation. *In* 'Proceedings of the 19th ACM International Conference on Information and Knowledge Management'. 2.2.1, 2.4, 4.1, 4.2.1

Yue, Y., Gao, Y., Chapelle, O., Zhang, Y. & Joachims, T. (2010). Learning more powerful test statistics for click-based retrieval evaluation. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.1, 2.3.2, 6.1, 6.2.1, 6.2.2, 6.4, 6.4, 6.4, 6.7, 6.7, 6.7, 6.9, 6.10.1, 7.3, 8.4.2.1, 8.5, 8.6, 9.2, 9.3

Zhang, A., Goyal, A., Kong, W., Deng, H., Dong, A., Chang, Y., Gunter, C. A. & Han, J. (2015). adaQAC: Adaptive query auto-completion via implicit negative feedback. *In* 'Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 21, 4.2.4

Zhang, Y., Chen, W., Wang, D. & Yang, Q. (2011). User-click modeling for understanding and predicting search-behavior. *In* 'Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'. 4.7.1