University of Copenhagen

Online Evaluation of Rankers Using Multileaving

Author:
Brian Brost

Supervisors:
Professor Ingemar J. Cox,
Associate Professor
Christina Lioma,
and
Associate Professor
Yevgeny Seldin

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Department of Computer Science

July 6, 2017

Abstract

This thesis deals with two central challenges in the online evaluation of rankers for information retrieval: (i) The design of multileaving algorithms and (ii) how best to manage the exploration-exploitation tradeoff associated with online evaluation using multileaving.

Multileaving is an online evaluation approach where the ranked lists produced by a set of rankers are combined to produce a single ranked list which is presented to users of a system. The quality of the rankers is then inferred based on implicit user feedback, i.e. which items the user clicks on. Multileaving is a generalization of interleaving, which differs from multileaving in only combining pairs of rankers at a time. Multileaving has been shown to reduce the quantity of feedback needed in order to evaluate rankers relative to interleaving.

We show that prior multileaving methods can be much less accurate than previously believed. In particular, prior multileaving methods fail to account for the interaction between how they create the multileaved lists presented to users, and how they use implicit feedback to infer the relative qualities of rankers. This can result in the quality estimates of prior multileaving methods depending on artefacts of the multileaving process, rather than the quality of the rankers being evaluated.

We introduce two new multileaving algorithms. Sample-Only Scored Multileaving (SOSM) is the first multileaving algorithm to scale well with the number of rankers being compared, without introducing substantial errors. Multileaving using Importance Sampling (MIS) is the first multileaving algorithm for which we can provide provable guarantees regarding the accuracy of evaluation.

Multileaving evaluates a chosen set of rankers. However it does not address how to choose these rankers. This is a classic exploitation versus exploration problem. On the one hand we would like the multileaved list to contain relevant documents, i.e. we should exploit the rankers we believe are good. On the other hand, other rankers may be better, i.e. we should explore rankers we are uncertain of. This problem has been previously framed as a dueling bandit problem when evaluating rankers using interleaving.

We extend the dueling bandit framework to managing the exploration-exploitation tradeoff associated with most currently existing multileaving algorithms. This is designed for algorithms where the outcome of the multileaving is a binary outcome, reflecting whether one ranker was better than another in a comparison. For this setting we introduce multi-dueling bandits, and show that regret can be reduced by orders of magnitude relative to what is attainable for dueling bandits.

For managing the exploration-exploitation tradeoff associated with multileaving algorithms such as MIS, which output absolute scores for rankers, we introduce a new variant of the bandits with multiple plays setting. This distinguishes itself from previous multiple play settings in that the number of arms to be played at each iteration is not fixed. Thus, it is possible to converge on exclusively selecting the best arm.

Dedication

To Declan,
Ar dheis Dé go raibh a anam

Acknowledgements

I can't thank Ingemar, Christina, and Yevgeny enough for their guidance and support over the last years, and for the patience and faith they've shown me throughout. I'd also like to thank all my other colleagues and fellow PhD students, in particular Noy, Casper, and Niels, who helped make my time at DIKU such a fun and positive experience. Finally, I'd like to thank my family, my girlfriend, and my friends for always being there for me, and for putting up with me when I got grumpy.

Contents

Chapter 1

Introduction

1.1 Online Evaluation in Information Retrieval

Ranking is a central problem underlying many of the most important services on the web today. Examples of applications include the need to rank search results for a web search engine or to rank products for recommendation in e-commerce. This thesis explores the use of multileaving for the online evaluation of ranking algorithms in information retrieval (IR). This section explains why evaluation of ranking algorithms, also called rankers, is a central problem in IR and related fields; why online methods are such an important part of ranker evaluation; and finally, why multileaving methods have the potential to solve many of the problems faced when attempting to evaluate rankers online.

Since ranking is such a central task for many applications, engineers and researchers make considerable efforts to develop and optimise ranking algorithms. Furthermore, because of the huge volumes of users on many web services, the potential impact on revenue of even very small improvements in ranker quality can be immense. It is therefore of critical importance to be able to reliably evaluate even very small differences in ranker quality. There are most often few theoretical guarantees that a particular ranking algorithm is good, so the cornerstone of ranker evaluation is thorough and careful empirical validation [36].

Ranker evaluation methods can be divided into three broad categories: Offline, online, and lab-based studies.

1.1.1 Offline Evaluation

Offline evaluation was long the dominant form of evaluation used in IR, and most commonly followed the Cranfield paradigm [96]. Here the starting point is a test collection; consisting of queries describing some information needs; documents, which can be web pages, tweets, images etc.; and relevance judgements, explicit judgements of how relevant the documents are to the given information need. Given this test collection we can then evaluate a variety of rankers according to various metrics most often measuring how highly the rankers place documents judged relevant. Relevance judgements, and the associated metrics, will be described in more detail in Sections 2.2.1 and 2.2.2 respectively. This approach provides a repeatable experimental setup on which experiments can be run quickly once the test collection has been created, and for which it is simple to carry out statistical tests to verify whether observed results are significant.

A key limitation of this experimental setup is the reliance on relevance judgements. Human judges have to estimate the relevance of documents to

often ambiguously described information needs, which may be unrepresentative of the information needs of real users. This can be expensive, and cannot be done exhaustively for large collections. Furthermore it can be difficult for judges to decide which documents are really useful to the end user. In particular, since the judges are often not involved in authoring the queries, they may themselves misunderstand the information need represented by the queries. Furthermore, a judge can only guess at the utility of a document in a personalised search setting, and judging cannot be easily updated to take into account new information, something that may be highly important for news retrieval. Finally, although the metrics used in offline evaluations measure certain notions of ranking quality, improvements in these metrics do not correlate consistently with increases in user satisfaction [96] or system revenue, which can often be the ultimate standards by which we wish to evaluate a ranking algorithm.

1.1.2 Lab-based Studies

Lab-based studies involve inviting users to participate in controlled experiments, where their experiences can be evaluated explicitly [10]. Unfortunately, although lab studies can provide feedback which is difficult to obtain otherwise, they are expensive, problematic to replicate and scale up, and since the number of users is generally small, it can be difficult to obtain a representative sample of the entire population of users for which the evaluation is meant to hold. A further limitation of lab-based evaluations is that participants may behave differently in a laboratory environment to how they would in more natural use cases [62].

1.1.3 Online Evaluation

Online evaluation of rankers is the evaluation of rankers in a *fully functioning* system based on *implicit* measurement of *real users'* experiences of the system in a *natural* usage environment [53].

By evaluating in a natural usage environment, i.e. based on how people use the system in their day-to-day lives, we can avoid a key problem of offline evaluation methods, that they can only approximate a real user's feedback, and we can avoid the possible distortions that can occur due to the less natural usage environment present in a lab-based study. Furthermore user behaviour can be easily logged with no additional effort from the user. This provides online evaluation methods with inexpensive access to large amounts of timely training data [26]. On the other hand, the implicit measurement of feedback, meaning the logging of clicks and other user behaviours, is noisy and difficult to interpret. As a result, these large amounts of training data are necessary to reliably infer quality differences between rankers. A click during a web search session can be a mistake, or even if it is not a mistake, cannot be interpreted as an absolute signal of quality. Instead, a click on a given document may only support the relative judgement that this document was more useful than the other documents inspected by the user.

One of the key drawbacks of online evaluation methods is that the outputs of new, potentially poor, rankers are presented to actual users. If a new ranker is poor, users will be presented with poor results and, in the worst case, might abandon the service [54]. Conversely, if new rankers are not

presented there is a risk of overlooking better rankers in the pool of rankers. In online learning the question of determining a proper exploration level is known as the *exploration-exploitation tradeoff*, an issue we will return to in Section 1.1.4, and which will be a major focus of this thesis.

The gold standard for online evaluation of rankers is *A/B testing* of the rankers on separate random subsets of the users or queries [99]. A/B testing allows for rankers to be compared on real users, according to the exact, specific use case that the experimenter wishes to examine, and according to the exact metric by which the experimenter measures success. The primary cost associated with A/B testing is the number of user impressions that are required to reliably distinguish performance. Since we can measure exactly what we want with A/B testing, the goal of alternative online evaluation methods should be to replicate the expected outcomes of A/B tests, while requiring fewer user impressions than A/B testing.

In online evaluation, it is often easier for users to make relative judgements, rather than absolute judgements. For example, it is easier for a user to say that document A is more relevant for a certain query than document B, than to say how relevant each document is. This intuition partly motivates the introduction of *interleaving* as a method to compare rankers. Interleaving methods have two stages, and compare pairs of rankers by first combining the ranked lists produced by each ranker into a single ranked list and displaying this list to the user. They then infer which ranker is better from implicit feedback, e.g. clicks, collected from the user. This approach has the benefit that the comparison is carried out on the same user, eliminating the between user variance which would affect a comparison between rankers A and B on separate users. Interleaving methods were found to require 1-2 orders of magnitude less interaction data than absolute metrics to detect even small differences in retrieval quality [26]. Additionally, it has been shown that the credit inference stage of interleaving methods can be tuned so that their outcomes agree well with the relative outcomes of A/B testing [99].

Multileaving is a generalisation of interleaving that allows more than two rankers to be simultaneously compared [102, 98, 17]. In this case, $K > 2$ rankers are compared by creating a new ranked results list that consists of documents selected from the documents retrieved by the K rankers and then inferring based on the user's clicks how good each ranker is. Multileaving has been shown to use click feedback more efficiently than interleaving [102].

Like interleaving, multileaving methods have two distinct stages; the first stage involves sampling the documents to be displayed to the user, and the second stage assigns credit to the rankers based on the user's clicks. The sampling stage is often a straightforward generalization of those proposed in the interleaving literature, for example one method is to randomly order the rankers and then, in turns, sample the top remaining document from each ranker. These sampling strategies are not uniform, i.e. some documents are much more likely to be sampled than others. One of the main contributions of this thesis is to show that for multileaving, the expected outcome of the credit assignment stage is affected by the probabilities of the documents being sampled during the first stage. Specifically, the ranker quality estimates in multileaving methods are skewed by artefacts of the sampling process, and this can cause substantial errors in the accuracies of multileaving estimates of ranker quality.

Multileaving offers dramatically improved efficiency over interleaving, allowing large numbers of rankers to be compared with very little interaction data, however this comes at the price of the above described problem which can affect the accuracy of the comparisons. One of our main contributions will be to provide a multileaving algorithm which solves this problem using importance sampling.

1.1.4 Exploration-Exploitation Tradeoff

Finally, we return to what we earlier described as one of the main issues when attempting to carry out online evaluation of rankers, the risk of displaying results from suboptimal rankers to the user. Interleaving and multileaving address the problem of how to compare rankers simultaneously, but do not address the critical question of what rankers to choose for each comparison, i.e. to resolve the exploration-exploitation tradeoff. This is critical since we would like to minimize how often we show poor results to the user even as we evaluate rankers of uncertain quality. *Dueling bandits* is an elegant mathematical framework that provides a principled way for dealing with the exploration-exploitation trade-off in learning with relative preference feedback from pairwise comparisons [123]. It has been successfully applied to online ranker evaluation based on interleaving [132, 133, 131]. Prior to the work contained in this thesis, there was no equivalent to the dueling bandits framework for multileaving. We describe our extensions of dueling bandits for use with multileaving in the next section.

1.2 Main Contributions

In this thesis we explore the problem of online evaluation using multileaving. Our main contributions can be separated into multileaving contributions and contributions to managing the exploration-exploitation tradeoff faced when carrying out online evaluation of rankers using these multileaving methods. The multileaving contributions are two multileaving algorithms and the discovery that a key property of the multileaving setting causes prior multileaving methods to be error prone. The contributions to managing the exploration-exploitation tradeoff are two corresponding algorithms for managing this tradeoff with two different classes of multileaving algorithms. We begin by describing our multileaving contributions.

Sample-Only Scored Multileave (SOSM) [17] is the first multileaving method to scale well with the number of rankers being compared, without introducing substantial errors. The central idea of SOSM is that rankers are evaluated only on the basis of their relative rankings of the documents included in the multileaved ranking. This means that each recorded click can be used to infer something about the relative quality of each pair of rankers, but no ranker is unfairly advantaged by having more of its highly ranked documents in the multileaved list. This efficient use of feedback allows SOSM to estimate the relative qualities of rankers, even after very few comparisons, regardless of how many rankers are being compared. SOSM was found to be substantially more efficient than prior multileaving methods, with prior methods often requiring twice as many user interactions to obtain similarly good ranker quality estimates [17].

The main non-algorithmic contribution of this thesis is that we demonstrate that multileaving methods need to properly account for the probability of a document being presented to the user in order to score rankers correctly. Prior multileaving methods, including SOSM, did not properly account for this, and consequently, and contrary to initial evaluations of multileaving methods, they are prone to being inaccurate. Concrete examples of how this affects the various prior multileavings methods are provided in Section 2.3.5.1.

The second multileaving algorithm contributed as part of this thesis, Multileaving using Importance Sampling (MIS), is the first multileaving algorithm to properly account for the probabilities of documents being presented to the user when scoring rankers. This is therefore the first multileaving method, which can reliably be used to accurately estimate the quality of rankers. MIS was shown to be highly accurate, scale well with the number of rankers being compared, and was shown to reliably evaluate rankers according to many different important information retrieval metrics. Finally, MIS is the first multileaving method that can be used in an unbiased manner on historical interaction data. This is possible since the importance sampling in the scoring function allows MIS to compensate for the difference in probability that a given document was presented to the user in the historical data, and the probability that the document would be presented to the user if a new multileaving was performed.

Since online evaluation of rankers involves displaying results from potentially inferior rankers to end users, a critical task is to balance the resulting exploration-exploitation dilemma when employing multileaving methods. Our main contributions in this area are given below.

We extend the dueling bandit framework and propose a *Multi-Dueling Bandit* algorithm that provides an intelligent selection of rankers for simultaneous comparisons based on upper confidence bounds, and improves the trade-off between exploration and exploitation[18]. This MDB setting assume binary relative scores for each pair of rankers being compared at each iteration, and is therefore applicable to use with multileaving algorithms such as SOSM, but not MIS, which produces non-binary scores for rankers at each comparison. With the MDB algorithm, performance, in terms of how many results from suboptimal rankers are displayed to users, was found to scale much better with the number of rankers being compared, and can be orders of magnitude better than that obtained by prior dueling bandit algorithms.

Our final contribution is a new *Bandits with Multiple Plays* setting applicable to absolute scores, such as those produced by MIS. Our algorithm for this setting performs similarly to our MDB algorithm, and give orders of magnitude improvements in performance over state-of-the-art dueling bandit algorithms.

In summary, this thesis advances the state of the art in multileaving to the point where multileaving methods combine efficiency and accuracy. Together with our work demonstrating how the exploration-exploitation trade-off can be handled using multileaving, this provides a promising avenue of future research into new applications of multileaving. Although the results presented in this thesis are promising, the experiments were all carried out on simulated users. This has become the standard in the interleaving and

multileaving literature, however more work is required to verify to what extent the findings on simulated users agree with evaluations on real users.

1.3 Thesis Overview

We have just introduced and motivated the main topics of this thesis, and given an overview of our main contributions. In Chapter 2 we provide the necessary background material to understand the main parts of the thesis. In particular, Sections 2.3 and 2.4 describe the material and related work necessary to understand and contextualise our multileaving and bandit contributions respectively.

Chapter 3 introduces our new multileaving algorithms and provides a thorough evaluation of these algorithms relative to the prior state of the art in multileaving. Chapter 4 introduces our corresponding new bandit settings and algorithms for managing the exploration-exploitation tradeoff associated with online evaluation using multileaving algorithms.

Finally in Chapter 5 we summarise our main findings and present perspectives on potential areas of future work in online evaluation using multileaving.

The work in Section 3.2 is adapted from work published in [17]. The work in Section 3.3 has been submitted to CIKM 2017. The work in Section 4.1 is adapted from work published in [18].

Chapter 2

Background

2.1 Information Retrieval

Much of the research on ranking algorithms has been done within the context of information retrieval. We therefore provide a brief introduction to some of the most important concepts. Information retrieval (IR) deals with finding material, often text, but also many other forms of material, usually of an unstructured nature, that satisfies an in an information need from within large collections [79].

Thus the central task in information retrieval is finding information that satisfies an information need, and ranking this information according to some measure of how well it satisfies the information need. As noted in the introduction, decades of research and engineering have gone into developing better ranking algorithms to assist in this task, and these competing algorithms need to be evaluated in order to decide which to use for a given situation. A central problem, which can be the bottleneck in improving IR systems, is that the evaluation required to confirm a potential improvement can be time-consuming and costly [71].

2.1.1 Ranking Models

Since this thesis concerns the evaluation of ranking algorithms, we will now briefly introduce some of the key concepts in ranking within text information retrieval. In particular we will focus on ranking models, which model the assumptions we make about how ranking should be carried out, and which have played an important role in the development of many ranking algorithms [36].

Most early information retrieval systems, and many current email search systems [24], employ a boolean retrieval model. Here the retrieval task is considered as a filtering process. In response to a query 'jaguar NOT car', a boolean retrieval algorithm would return all those documents in the collection which contain the word 'jaguar' but do not contain the word 'car'.

The *boolean retrieval model* is attractive for its simplicity, and the fact that its outputs are predictable to the user. However many evaluations have found that boolean retrieval systems perform worse than algorithms which attempt to weight terms according to notions of importance or relevance [95]. Additionally, boolean retrieval systems can perform particularly poorly for inexperienced users, who either cannot properly formulate their queries, or struggle to understand what terms are useful for filtering through the collection [36]. Despite these apparent shortcomings some studies have found that users, particularly those that consider themselves experts, prefer

boolean retrieval systems [115, 52]. This apparent preference for boolean retrieval needs to be interpreted carefully however, in particular, since boolean retrieval systems were much more prevalent when those studies were carried out, and the user preferences might partly reflect greater familiarity. Note that even today, purely boolean retrieval can be important in areas such as patent retrieval, where recall is highly important [22].

The *vector space model* became the dominant model for ranking algorithms in the 1960's and 1970's [22]. Unlike the boolean retrieval model, it provides a simple framework for incorporating concepts like term weighting and ranking [36]. Rankings can be inferred from an ordering on the distances between vectors representing the query and the documents in the collection. Unfortunately, although providing a simple framework, the vector space model provides little guidance for choices of how to practically incorporate the various elements like term weighting, or choice of distance function into a ranking algorithm [36].

One of the most important concepts underlying many of the most successful ranking algorithms is *tf-idf* term weighting. Here *tf* stands for term frequency, the number of times a term occurs in a document and *idf* stands for inverse document frequency, the inverse of the number of documents in the collection containing a term [105]. The tf-idf weighting is a product of these two factors. This reflects the intuition that a document containing a term many times is more likely to be about the term than one which contains the term only a single time, and that a term which is contained in many documents might not be very discriminative for deciding what those documents are really about.

There have been many models that could be described as probabilistic in IR, but the *probabilistic model* for IR is usually used to refer specifically to methods arising from the work of Stephen Robertson, Karen Spärck Jones and their collaborators beginning in the 1970's [22]. The probabilistic model starts from a basic assumption that if an IR system ranks the documents in the collection according to their probability of relevance, the overall effectiveness of the system to its user will be the best that is obtainable [92]. Note that this assumption can be regarded as an oversimplification, since it ignores document dependencies such as the desire to present a diverse set of documents to the user.

Nonetheless it proved extremely successful, and one instantiation, *BM25* is one of the best performing non machine learned algorithms in IR [91]. BM25 combines a tf-idf weighting scheme with a document length normalisation to reflect the fact that a term which occurs a number times in a short document might reflect relevance to a greater extent then the occurrence of that term the same number of times in a long document, such as an encyclopaedia.

Language models are used in diverse fields such as machine translation, speech recognition and information retrieval [10]. A language model is a probability distribution over a sequence of words or phrases, which models how likely each word or phrase is to appear in a given text. They can be used for retrieval by first generating language models for each document and then seeing which document language model has the highest probability of generating the query string [35]. Since many words would have a zero probability of occurring in a language model based purely on the individual documents, language modelling techniques often employ smoothing

techniques which provide a transition between the document language model and the collection language model [127].

We have until now only discussed ranking models in terms of how they match documents to queries by considering things like term and document frequencies. This focus on the match between query and document ignores two other types of signal which might be important in IR: Query-independent properties of the document, and document-independent properties of the query. The fact that a document appears to be about the same topic as a query might be reflected in the term frequencies, but a web page could still be undesirable to a user because it is spam. More generally, the link structure of the web can be used to infer the importance or trustworthiness of a web page [59], for example in the PageRank algorithm [86]. A simple example of an important document-independent property of the query is that the location from which the query 'weather' is issued can be the main determinant of what results a mobile phone user is looking for.

An additional important consideration, identified very early in the history of information retrieval is the vocabulary mismatch problem [94, 34]. This is the problem that a single concept or thing can have several names. In recent years semantic search has become an increasingly important area of research in IR. Here semantic search means that rather than just trying to match query to topic based on the words of the query, we instead try to use the meaning of the query [47]. This has led to the development of extensions of vector space models [114], and language models which, for example, use word embeddings to circumvent the vocabulary mismatch problem.

We have described some of the most important models for text retrieval, but we have ignored the fact that many of the ranking algorithms derived from these models have parameters that need to be tuned in order to maximise their performance for a given application. BM25 has two parameters; models like BM25F [91] weigh different parts of a document differently, so that, for example, the terms in the title of a web page might be considered more important than those in the body of the web page; and a language model smoothing technique such as Jelinek-Mercer has a parameter controlling how much the document or collection frequencies of terms are weighted. Additionally, search engines may have many other useful signals such as the time a query was issued; demographic information or search history of the user who issued a query; the PageRank scores; and many other diverse sources of information that could be useful in the retrieval task. Many of these signals are complementary, and there is no obvious way to integrate them by hand, necessitating a different method of best constructing a retrieval algorithm.

One of the most powerful techniques to arise in information retrieval in recent years is that of using machine learning. *Learning to Rank* [78] is concerned with using machine learning to learn ranking functions. Most learning to rank methods have treated it as an offline supervised learning task. There are three main approaches to this type learning to rank, pointwise, pairwise and listwise approaches. Pointwise approaches attempt to learn the relevance of documents. These can then be sorted to produce a ranked list. Pairwise approaches learn to decide given a pair of documents, which of the two is more relevant, an approach which can also be used to infer a ranked ordering on the documents. Finally, listwise approaches attempt to directly optimise the types of ranking metrics we will introduce in

Section 2.2.2.

A different approach is to attempt to use online click feedback to carry out online learning to rank. We will return to this topic after we have more thoroughly discussed the use of online click feedback in Section 2.3.1.

The important thing to note about these retrieval models we have introduced is that many of the decisions about parameters and about how to combine models have little theoretical motivation. Furthermore, to the extent that we can make theoretically well-founded decisions, these do not necessarily correlate with improved retrieval performance [36]. As a result, and in particular because of the growth of learning to rank approaches, the number of ranking models which need to be evaluated for a given application can be very large, necessitating inexpensive, scalable evaluation.

2.1.2 Personalization

Another trend that needs to be considered when deciding how to evaluate IR systems is the increased tendency of IR systems to employ elements of personalised search. For many applications, large potential improvements in search quality are possible because of the difference between what the average user is searching for, and what any one individual user is searching for [113].

Personalised search systems require some degree of personalised or situational evaluation, since the potential benefits of personalisation apply only on the individual level [39, 40]. In particular, this suggests that an evaluation based on a user's own clicks should be preferable to attempting to evaluate based on other people's relevance judgements.

2.1.3 Aggregated Search

An important element of modern web search engines is that they aggregate different types of results into a single results page that is displayed to the user [82]. This *aggregated search* can combine elements of relevant news, images, videos, as so called *verticals* in addition to the main results [3].

This presents additional challenges for IR evaluation methods, since these often assume a simple ranked list layout [128]. We will see in the coming sections that this presents a problem for both offline and online evaluation methods.

2.1.4 Fundamentals of IR evaluation

Before going into the details of evaluation methodologies, we begin by considering some more fundamental questions regarding IR evaluation. Returning to our definition of IR, we can regard information retrieval evaluation as the process of assessing how well a system meets the information need of its users [118]. An important consideration here is that the individual components of the system can be less important than its overall performance [34]. As a result, it may sometimes be desirable, as far as possible, to evaluate ranking algorithms and other components of IR as they appear within those systems, rather than as separate, independent components.

IR systems often face a tradeoff between two competing requirements, the needs for *effectiveness* and *efficiency*. Effectiveness can be regarded as including all aspects of the quality of the information presented to the

user, and how well the user's information need is satisfied, whereas efficiency concerns the system's resource consumption. The efficiency of a system can be roughly broken down into three components; time efficiency, i.e. how fast does the system serve its results; space efficiency, i.e. how much memory or disk space does it require; and cost efficiency, i.e. how expensive is the system to set up and operate [22]. For this thesis we restrict our attention to evaluating ranking quality, an aspect of IR system effectiveness.

The interplay between effectiveness, efficiency, and user satisfaction is not straightforward, with improvements in one aspect not necessarily correlating with improvements in user satisfaction [44, 2]. As a result, in the ideal case, evaluations cannot just focus on verifying that a change results in improved effectiveness, but should also examine to what extent the improvements in effectiveness translate to improved user satisfaction.

Traditionally, a fundamental concept in the study of IR system effectiveness has been the idea of *relevance* [32, 88]. There is no unanimously accepted precise definition for what relevance really is, but it can be regarded as capturing to what extent a document retrieved by an IR system satisfies the information need of the user.

Most traditional measures of effectiveness are based on how well a system retrieves the relevant documents in a collection in response to a query issued to the system. Two of the simplest and most important measures are *precision* and *recall*. The precision of a system is the proportion of the retrieved documents that are relevant, and the recall is the proportion of the relevant documents retrieved by the system. Other important metrics such as normalized discounted cumulative gain and mean reciprocal rank will be introduced later, but for now we return our focus to relevance. More specifically, in order to compute these metrics we first need to decide which documents are relevant.

This has traditionally been done by appointing judges to assess how relevant each document in the collection is to some query. Unfortunately this is prohibitively expensive to do exhaustively for even moderately sized collections, since it would require the relevance of each document in the collection to be assessed against each query [10]. Instead, the top-k documents retrieved by a group of reference rankers can be combined into a smaller document *pool* for each query, and these documents can be assessed. Documents not included in the pool are assumed to be irrelevant to the query. This approach, part of the Cranfield paradigm [118], has several potential weaknesses [129] which we will discuss in more detail in Section 2.2.1.

Alternatively, systems could try to obtain explicit relevance judgements directly from users, however this can be disruptive to the user experience, and this approach is often considered of limited utility [63]. Instead, user's actions can be interpreted as implicit feedback regarding the quality of the displayed results. Note that we cannot consider clicks on documents as direct relevance judgements, since this implicit feedback is noisy, and prone to bias which will be discussed in more detail in Section 2.3.1.

2.2 Offline Evaluation of IR Systems

We briefly discussed offline evaluation, particularly that based on the Cranfield paradigm, in the previous sections. We now consider some of the components and strengths and weaknesses of offline evaluation methods in more detail.

Recall that under the Cranfield paradigm the starting point is a test collection [96]; consisting of queries describing some information needs; documents, which can be web pages, tweets, images etc.; and relevance judgements, explicit judgements of how relevant the documents are to the given information need. Given such a test collection we can then evaluate a variety of rankers according to various metrics most often measuring how highly the rankers place documents judged relevant. We will begin with a more detailed discussion of the relevance judgements, since creating them presents the biggest cost associated with Cranfield-style offline evaluation.

2.2.1 Relevance Judgements

Relevance judgements have traditionally required human judges to estimate the relevance of each document relative to a query representing some information need. These relevance judgements may be binary, indicating only if a document is relevant or not, or graded, with, for example, a relevance judgement of 4 corresponding to a highly relevant document, and decreasing levels of relevance up to a relevance of 0 for a document which is not relevant at all.

Perhaps the most well known problem with this setup is that for large collections it is impractical to create relevance judgements for all the documents in the collection for each query. As a result, for example, the top 100 documents retrieved by some set of rankers would first be collected into a document *pool*, and only the documents in this pool would then given relevance judgements. The unjudged documents outside this pool would be considered not to be relevant. The obvious problem with this approach is that a ranker which retrieves relevant documents outside of this pool risks having its performance underestimated by any metric using these relevance judgements. This problem is known as *pool bias*, and is exacerbated by increasing collection size, even if the size of the document pool is increased too [20].

Early research suggested that this problem was negligible in practice [50, 129]. For example in [129] the rankers used to create the document pool were assessed against the relevance judgements obtained when the documents contributed by that ranker were excluded from the pool, and the relative performances were found to be robust to this change. However later work has suggested that this problem can substantially bias evaluations [20, 23], and this bias can be particularly strong for systems radically different from those used to create the document pool. An example of this could be that a ranker which finds relevant documents based on semantic, rather than term matching, would not be rewarded in the evaluation if the pool was created purely by rankers which carry out term matching.

Since pool bias can be somewhat mitigated if the size of the document pool is increased, it is important to be able to generate relevance judgements in a cost-efficient and scalable manner. A promising tool for generating

relevance judgements is crowd sourcing. Unfortunately it is well known that crowd sourcing can be unreliable because of the non-expert nature of participants, and the fact that their incentive is often to use the minimal effort required [84]. Despite these risks, research suggests that although individually less reliable than expert annotators, in aggregate crowd sourcing can produce high quality relevance judgements [84, 14]

There are more fundamental problems with relevance judgements, than just their incompleteness and the associated pool bias. Relevance judgements generally assess documents against some notion of topical relevance, i.e. if the documents are about the same topic as the information need, but such judgements may not sufficiently weigh factors such as utility [32] or diversity, and may not even be able to address a concept like personal or situational relevance. Although specialised or graded relevance judgements may approximate some of these factors, an unavoidable problem for any relevance judgement is that certain aspects can only be assessed by the users themselves [81].

Finally, we can try to interpret implicit feedback generated from click-through data as relevance judgements. This is problematic, since clicks are biased in a number of ways. They also suffer from a form of pool bias, since documents not displayed to users cannot be clicked on. Furthermore there are other biases which will be discussed in more detail in Section 2.3.1. As a result, it is difficult to interpret clicks as judgements of a document's relevance in an absolute sense. However they can be interpreted as relative relevance judgements between documents in a reasonably accurate manner [58]. For example, a click on a document reliably suggests that it is more relevant than a more highly ranked document which was not clicked.

2.2.2 Metrics

Given relevance judgements, a variety of metrics are used to assess the performance of rankers. We will introduce some of the most important metrics before discussing their relation to user satisfaction.

The precision of a system is the proportion of the retrieved documents that are relevant, and the recall is the proportion of the relevant documents retrieved by the system. Often it is considered more important for a ranker to rank the top of its ranked list well than it is to assess how good the documents ranked at lower positions are. This can be justified by the fact that users are much more likely to actually look at the top ranked documents [33]. Two rankers which each rank a single relevant document at ranks 1 and 100 respectively would have the same precision, even though intuitively, the performance of the first ranker seems much better.

This motivates two variants of precision, *Precision @ K* (P@K) and *Average Precision* (AP). P@K is defined as the proportion of documents retrieved in the top K which are relevant, and

$$AP = \sum_{k=1}^{n} \frac{P@k \times \delta(k)}{|R|} \tag{2.1}$$

where n is the total number of retrieved documents, R is the set of relevant documents and $\delta(k)$ is the relevance score of the document if the document ranked at position k. This relevance score is 1 if the document is relevant,

and 0 otherwise. Note that the cutoff version of Precision, P@K still suffers from the problem that the order of items does not affect the score, as long as they are contained in the top K. Conversely, AP rewards a ranker more the higher it ranks relevant documents.

For certain types of use cases we might only be interested in how highly a single item is retrieved, for example if a query is issued for a specific website. For this type of navigational query, one possible metric could be $P@1$. However this can be unstable, and would not distinguish between a ranker which ranked the correct document at position 2, and therefore only narrowly failed, and another ranker which placed that same document at position 100. In such a scenario a more appropriate metric might be the so-called *mean reciprocal rank* (MRR). The mean reciprocal rank over a set of queries Q is

$$MRR = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{rank(q)}, \tag{2.2}$$

where $rank(q)$ is the rank of the first relevant document for query q.

The metrics we have described so far cannot take into account graded relevance. We now introduce the *normalized discounted cumulative gain* which has many variants. We will use a popular variant [61, 120] used in Lerot [100], the software package which we have used for our experiments. We first need to introduce the *cumulative gain at K* and *discounted cumulative gain at K*, given by

$$CG@K = \sum_{k=1}^{K} 2^{\delta(k)} - 1 \tag{2.3}$$

$$DCG@K = \sum_{k=1}^{K}\frac{2^{\delta(k)} - 1}{log_2(k+1)} \tag{2.4}$$

where $\delta(k)$ is the graded relevance score of the document ranked at position k. Then we can define

$$nDCG@K = \frac{DCG@K}{iDCG@K} \tag{2.5}$$

where $iDCG@K$ is the $DCG@K$ score obtained if the documents are ranked according to their graded relevance. $nDCG$ is a popular IR evaluation metric for graded relevance judgements [61, 120].

Surprisingly, given the widespread popularity of nDCG, user satisfaction with the Google search engine has been found to be more strongly correlated with $CG@10$ and $P@10$ than $DCG@10$ and $nDCG@10$ [2]. This finding was subsequently contradicted in a much larger user study [97], which found that $nDCG@10$ was more strongly correlated with user preferences than the aforementioned metrics.

These somewhat contradictory results underline the fact that there is a complicated interplay between offline measures of effectiveness and user satisfaction [44, 2, 97]. In light of this complicated interplay, it is important to empirically validate, through either online evaluations or user studies, the findings of offline evaluations [60].

2.2.3 User Studies

Lab-based studies involve users participating in controlled experiments, where their experiences can be evaluated explicitly [10]. This provides a way of validating the results of offline evaluations. However as already noted, there are aspects of system effectiveness, particularly when dealing with personalised search results, that can only be evaluated by the end users themselves [81].

Despite this, user studies can provide insights into what causes a system to be perceived as effective or ineffective, such as why a user prefers one result over another, that cannot be obtained by normal offline evaluations, or through online implicit feedback. Additionally, user studies can be valuable tools for validating the results of offline and online evaluations. However, higher throughput evaluation methods are required for experimenting on the hundreds of variants of their rankers that commercial search engines might wish to evaluate at any given time [70].

A significant reason for this is that lab-based studies are expensive, difficult to replicate and scale up, and since the number of users is generally small, it can be difficult to obtain a representative sample of the entire population of users for which the evaluation is meant to hold. An additional limitation of lab-based evaluations is that participants may behave differently in a laboratory environment to how they would in more natural use cases [62].

2.3 Online Evaluation of IR Systems

Online evaluation of rankers is the evaluation of rankers in a *fully functioning* system based on *implicit* measurement of *real users'* experiences of the system in a *natural* usage environment [53]. The most common form of implicit user feedback used in online evaluation is click feedback, and we will discuss how clicks can be interpreted, how we can model user click behaviour, and some of the most important online evaluation methods in the coming sections.

2.3.1 Interpeting Click Feedback

Logged implicit user feedback, e.g. clicks, has the potential to give information about the effectiveness of search results. The fact that a user clicks on a document could be taken as an explicit judgement that the given document is relevant to the user. Unfortunately, user behaviours are noisy and prone to a variety of biases which need to be accounted for [58, 90] when interpreting clicks.

Position bias is the tendency of users to be more likely to click on documents ranked in a high position than those ranked lower [58]. This can have a number of different causes, including that users tend to read the results page from top to bottom and might stop before ever considering a document with a low position. Additionally users might trust the search engine sufficiently to assume that higher ranked documents are more relevant, and might therefore click on a document purely because of their trust in the search system.

Joachims *et al.* showed that the quality of the ranking influences the user's click behaviour [58], with users more likely to click on less relevant

documents if the overall quality of the ranked list is worse. This makes the interpretation of clicks as an absolute relevance judgement problematic. It may be difficult to interpret clicks as absolute judgements, however clicks can be interpreted as relative relevance judgements reliably [58, 90]. For example, a clicked document at position 2 in a ranked list can be reliably interpreted as being more relevant than an an unclicked document at position 1.

Another potential source of bias is *presentation bias*. Search engines produce small snippets summarising parts of their retrieved documents and users have been found to be more likely to click on documents with attractive snippets, regardless of the actual relevance of the document [126]. This can be particularly problematic from the point of view of preventing the ranking of spam websites, since they might attempt to manipulate their snippets to be unrepresentative of their real content.

Selection bias [119] is the phenomenon that the collection of queries for which clicks are recorded is biased compared to the entire collection of queries. This means that certain queries will occur less frequently in the click log used for online evaluation than in the overall query log. An example of this could be that queries asking about the weather are underrepresented in the click logs, relative to how frequently they are actually issued, because the user is presented with information directly in response to the query, and does not need to click on any links.

Although clickthrough data is an important source of implicit logged feedback, it is not the only type of feedback we can infer from search logs. An example of another type of behaviour that can be used is *abandonment*, where a user issues a query, but does not click on any results. This could be considered a negative signal, since it may indicate that the user was not presented with any useful results to click on. Conversely, it can also reflect the fact that a user's information need was met directly on the search page as in the above weather query example, or if the information need is met by one of the presented snippets. This type of *good abandonment* [76] can be partly identified, for example by tracking cursor movement [57], or by checking if the query was subsequently reformulated [51].

As a final example of a form of implicit feedback which could be used, although it is not currently logged by search engines, eye tracking has been found to be predictive of relevance [93, 46, 21].

Ranker evaluation methods which use click feedback must be carefully designed to ensure that the above described biases do not impact rankers unequally, and we will see in Sections 2.3.3, 2.3.4 and 2.3.5 that controlling for bias is a central consideration when designing evaluation methods.

2.3.2 Click Models

If we can accurately model user click behaviour we can use these models to design ranker evaluation methods which can control for the user biases. Furthermore if the user models are accurate, simulations can provide valuable insights into how well ranker evaluation methods perform, without having to experiment with them on real users. Click models can also help motivate the design of IR metrics [31].

As already noted, position bias is a key source of bias in clicks where the presentation order of documents affects their likelihood of being clicked on.

This problem is addressed by two of the most well known families of click models: position based, and cascade click models [33].

Position based click models decompose the probability of a click into the probability of looking at a document, given that it is at a given position, and the probability that the document is clicked, given that it was looked at. This can be regarded as a model where a user inspects the ranked list from top to bottom, and has a relevance-dependent probability of clicking on each document, and a fixed probability of stopping their examination of the documents. Cascade click models instead let the stopping probability also be dependent on the relevance of the document. This reflects the intuition that if a user's information need is fully satisfied by the first document encountered, there is no need to look at any other documents. Cascade click models have been found to significantly outperform position based click models [33].

Cascade click models are very popular and have been used to simulate experiments in most recent papers on interleaving and multileaving methods [55, 102, 98, 17]. Work on more systematic simulations of information retrieval interactions is relatively recent but promises to provide more realistic simulations [64, 9, 80].

Other, more advanced, click models which attempt to account for position bias include the dependent click model [49], the user browsing model [43], a dynamic bayesian network model [27] and the click chain model [48]. A survey of click models for web search that also account for important aspects of search such as aggregated search, diversity and multiple sessions is given in [28]. Recently, recurrent neural network and LSTM's have been used to learn click models directly from historical user interactions, without first assuming that the user inspects the results page in a particular way [15].

2.3.3 A/B Testing

An A/B test, sometimes also called a randomised, or controlled experiment, in its simplest manifestation, is an online experiment where users are randomly assigned to one of two variants of a system [71]. Relevant metrics can then be collected based on implicit feedback and statistical tests can determine if there are significant differences between the two variants. Usually the one variant of the system is the system as it is currently in operation and the other variant is a modification with a single change whose effect we wish to investigate. If the assignment of users between the variants is random, and the only change between the variants is what we wish to investigate [83], any difference in performance can be expected to be because of the change in the new variant, since this is the only consistent difference. For this reason A/B testing is considered the gold standard in online evaluation [71], and commercial search engines run hundreds of concurrent A/B tests on any given day [70].

A central focus in designing A/B tests is to select a good *overall evaluation criterion*, according to which it can be decided which variant performed better. Choosing this criterion is difficult because the criteria which might ultimately be of most interest, such as user satisfaction or long term system revenue, may not be directly measurable. We must therefore instead choose

proxies based on immediate signals like clicks, which correlate well with the long term goals [37, 71, 69, 41].

A/B tests can require hundreds of thousands of user impressions to detect small differences between systems [26]. Given the number of experiments that web service providers would like to carry out, the need to minimize possible deterioration in the experience of the user base when testing modifications is a limiting factor on how many experiments can be carried out. Another limiting factor, if either the user traffic is limited for a smaller web service, or if the number of desired experiments is huge, is that there might not be enough user impressions to test new variants as quickly as desired. For this reason, it is highly important to be able to use user feedback efficiently, and develop online evaluation techniques which require less traffic in order to determine if one system is better than another [112, 38, 68, 67, 65].

2.3.4 Interleaving

One of the reasons A/B testing can require so many user impressions to detect small differences in quality is that variants are not tested on the same user impressions. If we evaluate two systems using A/B testing, one system may initially encounter queries which are easier than those encountered by the other system. This can be expected to even out over a sufficiently large number of user impressions, but it presents an important source of variance in the metrics being studied. We now introduce a method which allows this source of variance to be eliminated.

Interleaving methods are an online evaluation approach where a pair of rankers is compared by combining the ranked lists produced by each ranker, displaying this combined list to a single user, and inferring which ranker is better based on clicks collected from the user. Interleaving methods thus have two stages, a document sampling stage where documents are sampled from the ranked lists of the two rankers into the joint list displayed to the user, and a credit inference stage where the rankers are given credit based on the clicks. State-of-the-art interleaving methods include Team Draft Interleaving [90], Probabilistic Interleaving [55], Optimized Interleaving [89] and Generalized Team Draft Interleaving [66]. Interleaving methods have been found to require 1-2 orders of magnitude fewer user impressions than A/B tests [26].

However, unlike A/B testing, which does not make specific assumptions about the presentation of the results, interleaving methods generally assume that the documents are presented in a single ranked list. Recently, interleaving methods have been developed to handle more general settings, including other layouts of the retrieved results, such as grid layouts [66], and have also been extended to enable the interleaving of aggregated search results [29, 30].

Finally, work has been carried out on how to optimise the credit inference stage of interleaving to agree with specific A/B testing user satisfaction metrics [99]. This is important since we saw in the previous section that there has been a lot of research on what metrics make sense for A/B testing, and we would like to be able to retain agreement with these metrics in the interleaving setting.

2.3.5 Multileaving

Schuth *et al.* generalised interleaving to simultaneous comparisons of more than two rankers, called *multileaving*, and introduced two multileaving methods: Team Draft Multileaving (TDM) and Optimised Multileaving (OM) [102]. TDM works similarly to team draft interleaving: the multileaved list is created in rounds, with each ranker contributing its highest ranked remaining document to the multileaving, up to a predetermined length, usually fixed to 10, to reflect the fact that users often only look at documents on the first page of the search engine results page. Note that in this case if more than 10 rankers are being compared, some rankers do not receive an opportunity to contribute a document. Rankers are then given a credit for each clicked document contributed by the given ranker, and a ranker is considered to have won the comparison over another if it receives more credit. OM generalises optimized interleaving [89], and attempts to construct a multileaving optimised for sensitivity, subject to certain constraints on the makeup of the multileaved list, for example to ensure that the best documents according to the individual rankers are used to create the multileaved list.

Before proceeding we note that we are primarily interested in three aspects of the performance of a multileaving algorithm: Firstly, how accurate they are relative to some desired ground truth. Secondly, how quickly does the multileaving method converge to varying levels of accuracy, i.e. how many user impressions are needed to get to an error rate of 10%. We will refer to this as the efficiency of the multileaving method. Finally, we are interested in how these first two aspects are affected by the number of rankers being simultaneously compared by the multileaving method. We will refer to this as the scalability of the method. A more detailed discussion of the problem setting for multileaving algorithms is given in Chapter 3.

Schuth *et al.* [102] found that TDM was substantially more efficient than interleaving methods, meaning that fewer user interactions were required to accurately determine pairwise preferences between rankers. They also found that TDM was substantially more accurate than OM, and that TDM was similar in accuracy to interleaving methods. Due to the poor accuracy of OM reported in [102] we will not consider OM further.

Since TDM credits rankers only for clicks on documents drawn from the corresponding ranker, TDM does not scale well to comparing more rankers than there are documents in the multileaved list. Since users of search engines typically only inspect the first results page, the multileaved list is effectively only of length 10. We are often interested in comparing significantly more than just 10 rankers, so there is a need for multileaving methods which scale better to larger comparison sets.

Schuth *et al.* [98] subsequently proposed Probabilistic Multileaving (PM), a generalization of probabilistic interleaving, which creates the multileaved list in a similar manner to TDM, although with a probabilistic sampling of documents. It then attempts to account for all the possible ways that a document could have been sampled into the multileaved list. This approach overcomes the limitation of TDM that clicks on documents only result in credit for a single ranker, limiting the scalability of TDM. PM was found to be substantially more efficient and scalable than TDM, while demonstrating similar accuracy. However, the experimental evaluations in [102, 98] were

only carried out on a very limited number of rankers, and results in [17] suggest that the findings do not generalise.

2.3.5.1 Shortcomings of prior Multileaving Methods

We now note flaws in the prior multileaving methods.

The multileaving methods we have discussed above can be prone to errors since they do not take into account the connection between the document sampling stage and the credit assignment stage. As a consequence, TDM can underestimate the quality of similar rankers, as the following simple example demonstrates: Consider the case of three rankers, A, B, C. Assume that rankers A and B have the same top ranked document, and that this is different from the top ranked document of ranker C. When creating a multileaved list of length 3, rankers A and B will be represented by either their top ranked or second ranked document. However ranker C will always be represented by its top-ranked document. Suppose that the top ranked documents for all three rankers are relevant, whereas their second ranked documents are not relevant. Then ranker C will outperform rankers A and B according to TDM, even though A, B, and C are equally good. TDM systematically underestimates the performance of rankers A and B. If the number of rankers that rank documents at similar positions increases, this underestimation of quality becomes more extreme. Thus, TDM does not properly account for the fact that in its credit assignment stage, similar rankers have to split the credit that they would receive from clicks on their highest ranked documents. This is particularly problematic since agreement between rankers can be a signal of good ranker quality.

Conversely, PM systematically overestimates the quality of rankers which are similar to the other rankers in the comparison set as discussed in [17]. To understand this flaw, we consider PM in more depth. Suppose we have a set $\mathcal{R}$ of rankers we wish to multileave. PM creates the multileaved list in rounds. In each round a random ordering of the rankers is decided. Then, a document is probabilistically selected from the next ranker in the ordering, where the probability of drawing a document d from ranker R is determined solely by the document's rank and is given by Equation 2.6, where $pos(d, R)$ is the rank of document d according to ranker R, and $\mathcal{D}$ is the set of documents ranked by R.

$$P_R(d) = \frac{\frac{1}{pos(d,R)^3}}{\sum_{d' \in \mathcal{D}} \frac{1}{pos(d',R)^3}} \tag{2.6}$$

Note that when a document is drawn, the document is removed from all the rankers' retrieved lists. In the next draw, the probabilities of the remaining documents are recalculated according to Equation 2.6, where the rankings, $pos(d, R)$, are now determined in the absence of previously chosen documents.

In the credit inference stage, PM considers all possible assignments of documents to rankers that could have occurred, i.e. from what ranker might the given document have been sampled, and weight each assignment based on its probability. A given assignment a of all the documents in the multileaved

list has probability, $P(a)$, given by

$$P(a) = \prod_{i=1}^{N} P_{R_{\alpha(i)}}(d_i) P(R_{\alpha(i)}) \tag{2.7}$$

where N is the length of the multileaved list and $P_{R_{\alpha(i)}}(d_i)$ is the probability of drawing document d_i from its assigned ranker, denoted by $R_{\alpha(i)}$. This probability is given by Equation 2.6, and $P(R_{\alpha(i)})$ is given by $1/|\mathcal{R}|$. For an assignment, a, Ranker R is given credit, $o_R(a)$ equal to the number of assigned documents clicked on. The total credit, $o_R(A)$, assigned to ranker R, is given by $o_R(A) = \sum_{a \in \mathcal{A}} o_R(a) P(a)$, where $\mathcal{A}$ is the set of all possible assignments.

PM overestimates the quality of rankers which are similar to other rankers in the comparison set. This is because they can benefit from the presence of documents contributed by similar rankers, and similar rankers will therefore perform better according to PM than they actually do in practice.

To illustrate this problem, consider the following simple example: three rankers and their corresponding retrieved lists: $(R_1 : d_1, d_2)$ $(R_2 : d_2, d_1)$, and $(R_3 : d_2, d_1)$. The possible multileavings of length two, are $\{d_1, d_2\}$ and $\{d_2, d_1\}$.

The former multileaving occurs with probability 0.37, and the latter occurs with probability 0.63. To see this, we note that if the first document is drawn from R_1, then according to Equation 2.6 the probability of d_1 being drawn first, corresponding to the first of the two possible multileavings is:

$$\frac{1}{\frac{1}{1} + \frac{1}{8}} = \frac{8}{9}$$

If the first document is drawn from R_2 or R_3, then according to Equation 2.6 probability of d_1 being drawn first is:

$$\frac{\frac{1}{8}}{\frac{1}{1} + \frac{1}{8}} = \frac{1}{9}$$

Taken together we see that the probability of the multileaving $\{d_1, d_2\}$ occurring is:

$$\frac{1}{3} \cdot \frac{8}{9} + \frac{1}{3} \cdot \frac{1}{9} + \frac{1}{3} \cdot \frac{1}{9} = 0.37$$

Suppose that d_1 and d_2 are both relevant and always clicked on, i.e. all three rankers have equal performance. Even though the rankers are equally good, R_1 will lose to the other two rankers with probability 0.63 due to the fact that when the multileaving $\{d_1, d_2\}$ occurs, R_1 is given more credit in the credit inference stage of PM, and if $\{d_2, d_1\}$ occurs, R_2 and R_3 are given more credit. Thus, in PM, the document sampling probabilities are higher for documents ranked highly by many rankers, but the credit assignment phase does not account for this.

Brost *et al.* proposed Sample-Only Scored Multileaving (SOSM) which creates the multileaved list in the same way as TDM, but gives credit to rankers based purely on how they would have ranked the sampled documents in the multileaved list, as opposed to using the documents' original positions in each ranking to determine their scores. This allows an arbitrary number of

rankers to be compared, overcoming a major shortcoming of TDM, without introducing the inaccuracy present in PM. Brost *et al.* demonstrated that SOSM was more efficient than previous multileaving methods, slightly more accurate than TDM, and substantially more accurate than PM [17].

Unfortunately, like TDM and PM, SOSM fails to correctly account for the relationship between its document sampling stage and the credit assignment phase. Specifically, documents from rankers which are similar to each other are more likely to be ranked highly in the multileaved list. Thus, if the user exhibits position click bias, this can result in the similar rankers being systematically favoured by SOSM over rankers which are dissimilar from the other rankers in the comparison set. Furthermore, it was shown in [18] that the estimates of SOSM can occasionally be distorted, in the sense that, depending on which other rankers are being compared in a given multileaving, SOSM can disagree with itself about the relative quality of pairs of rankers.

In Chapter 3.3 we propose using importance sampling to produce a multileaving algorithm which is unbiased. Note that in order to be unbiased, importance sampling requires that all documents have a non-zero probability of being sampled into the multileaved list in their position in the original ranked lists. As such, importance sampling cannot be used to correct either TDM or SOSM, since both algorithms use the same sampling algorithm which always samples the top remaining document from a given ranker, and therefore can have zero probabilities of sampling documents into their original positions according to some of the rankers.

2.3.6 Counterfactual Evaluation

While importance sampling has not been previously used for multileaving, it has been used for counterfactual evaluation. Counterfactual evaluation is a form of online evaluation in which the experimenter has (almost) no control over what is displayed to the user. Instead of interleaving or multileaving documents from multiple rankers, a single logging/production ranker is used but its result set may be randomly permuted. Counterfactual methods then attempt to answer the question, based on logged feedback for the logging ranker which is actually displayed to the users, of how a different ranker would have performed, had it been displayed to the users [16]. In order for this counterfactual evaluation to be unbiased, the above mentioned permutations of the logging ranker must have a non-zero probability of producing any ranked list that can be produced by the rankers that we want to evaluate [75].

Importance sampling, and variants thereof, have been used in this counterfactual setting [109, 110, 108, 77, 111]. To the extent that the assumptions made by the counterfactual evaluation method are met, and if the logging ranker is similar enough to the rankers being evaluated, counterfactual evaluation has many of the same advantages as interleaving or multileaving, without requiring full control of what documents are displayed to the user.

Some of the disadvantages of counterfactual methods are that the more dissimilar the rankers we wish to evaluate are to the logging ranker, the more randomisation, or logged data is required to reliably evaluate the rankers. In particular, increasing the randomisation in the results produced by the logging ranker can produce unacceptably poor result lists to be displayed to the end user.

2.4 Online Learning

Recall that one of the key drawbacks of online evaluation methods such as A/B testing, interleaving, or multileaving, is that we face a tradeoff between displaying the results that we believe are best to the user, or showing results which will help us learn more about the quality of the rankers we are evaluating, but at the cost of potentially showing poor results to the user [54]. Resolving this exploration-exploitation tradeoff can be formalised by different so-called bandit settings, depending on what online evaluation methods we are considering. We will now introduce the relevant literature corresponding to online evaluation using A/B tests, interleaving, and multileaving

2.4.1 Bandits

We can regard online evaluation when using A/B testing as a sequential decision making problem, where we face a stream of queries issued by users, and for each query we wish to choose which ranker's results to display to the user. This approach can be modeled within the multiarmed bandit framework [13], where we associate rankers with arms. At each iteration of a multiarmed bandit game the player picks one of K possible arms (in our case rankers) and observes the reward of that arm. Rewards of arms that were not selected remain unobserved. The goal is to select arms, so that the cumulative regret, defined as the sum of the differences across iterations between the reward of the unknown best arm and the reward of the chosen arm, is minimized.

2.4.2 Dueling Bandits

If instead of using A/B testing, we wish to use interleaving to compare rankers, we can instead regard our situation as a sequential decision making problem where we face a stream of queries issued by users, and for each query we wish to choose which two rankers to interleave to the user. This learning from relative feedback from pairwise comparisons can be modeled as a K-armed *dueling* bandit problem [123]. The goal is to select pairs of rankers, so that a mix of their rankings will be almost as good as the ranking of the best ranker in the pool. (We note that the model permits selecting the same ranker twice, as the first and the second element of the pair.) There are several possible definitions of the best ranker for pairwise comparisons. The most common definition is the *Condorcet winner*, which is a ranker that is (pairwise) better that any other ranker in the pool. Note that a Condorcet winner is not guaranteed to exist, for example in a situation with three rankers A, B, and C, where A wins in a comparison with B, B wins in a comparison with C, and C wins in a comparison with A. Although somewhat counterintuitive, this type of situation occurs frequently for interleaving methods [42].

The K-armed dueling bandit problem was introduced by Yue *et al.* [123], who also presented an algorithm called *interleaved filtering* (IF) for solving it. In IF, arms are eliminated sequentially by comparing them with the best currently known arm until they are defeated with sufficient confidence. Yue *et al.* [125] subsequently proposed an improved algorithm called *beat the*

mean (BTM). This algorithm attempted to reduce the number of comparisons needed by focusing on comparing the arm that had been used in the least number of comparisons with a randomly sampled arm that had not yet been eliminated. Yue *et al.* also presented experimental evidence confirming the superior performance of BTM over IF.

Zoghi *et al.* [132] proposed an algorithm for the dueling bandit setting based on the idea of *relative upper confidence bounds* (RUCB). The algorithm maintains a relative upper confidence bound on the probability that a given arm i is better than another arm j. The algorithm then selects an arm i that might be the best, based on its upper confidence bounds relative to all other arms, and then selects the challenger with the highest upper confidence bound relative to i. This approach was shown to outperform both IF and BTM. Zoghi *et al.* [131] subsequently proposed a divide-and-conquer algorithm, *MergeRUCB*, extending their earlier work in [132]. MergeRUCB was designed to perform well for problems involving a large number of arms. Extensive experiments by Zoghi *et al.* suggest that it substantially outperforms IF and BTM, and that for large numbers of arms it outperforms RUCB too.

Komiyama *et al.* [72] proposed an algorithm, *Relative Minimum Empirical Divergence* (RMED), which selects pairs of arms based on whether an arm has not been compared with other arms sufficiently often, or if it is not substantially beaten by many other arms. To decide if an arm has been sufficiently explored it uses bounds based on the KL-divergence. They showed that this algorithm outperformed RUCB and MergeRUCB.

Most recently, a new dueling bandits algorithm which selects both arms using Thompson sampling has been found to outperform prior dueling bandit algorithms [122].

Dueling bandits have been the focus of a lot of recent research, culminating in several new research directions and variants of the problem setting. We will briefly summarise some of these directions here.

Since the Condorcet winner is not guaranteed to exist, there has been work investigating algorithms given other concepts of winners, such as dueling bandits for a Copeland winner [130, 74] or for a von Neumann winner [42]. A Copeland winner is a ranker for which the number of pairwise comparisons won by the ranker is greater than the number of pairwise wins of any other ranker. A von Neumann winner is a probability distribution over the rankers, such that in expectation a ranker drawn from the distribution will defeat any other individual ranker.

It has also been shown that dueling bandits can be reduced to the standard multiarmed bandit setting [1]. Balsubramani *et al.* [12] provided the first algorithms which take advantage of the specific problem instance to produce instance dependent regret bounds. Finally, work on so-called adversarial bandits, where the rewards for each arm are not assumed to be identically and independently distributed, has been extended for dueling bandits by Gajane *et al.* [45].

The dueling bandit problem setting is limited to pairwise comparisons. We introduce a new problem setting, multi-dueling bandits [18], described in Chapter 4.1, for regret minimisation when simultaneously comparing an arbitrary number of arms. Subsequently, there has been further work on multi-dueling bandits [107]. However, this work by Sui *et al.* restricted attention to the case where the number of arms selected at each iteration is fixed. Sui *et al.* also produced earlier work on the closely related problem

of learning from subgroup rank feedback, although this problem setting distinguishes itself from our multi-dueling bandit setting in that the number of arms selected at each iteration is again fixed [106].

2.4.3 Bandits with Multiple Plays

In Section 4.2 we introduce a new bandits with multiple plays setting applicable, for example, to regret minimization when using a multileaving method like our new Multileaving using Importance Sampling algorithm introduced in Section 3.3.

There has been prior work on bandits with multiple plays [87, 116, 73], but prior work was again restricted to the case where the number of bandits selected at each iteration is fixed. This prior work is therefore not directly applicable to the online ranker evaluation problem, where we wish to eliminate arms as we become more certain that they are not optimal, rather than continuously selecting a fixed number of rankers.

Chapter 3

Multileaving for Online Ranker Evaluation

We first introduce our multileaving problem setting in Section 3.1. Then we present our proposed algorithms, Sample-Only Scored Multileave (SOSM) [17] in Section 3.2, and Multileaving using Importance Sampling (MIS) in Section 3.3. We present thorough experimental evaluations of SOSM and MIS in Sections 3.2.2 and 3.3.3 respectively.

3.1 Problem Setting

We are given a set $\mathcal{R}$ of rankers and a set $\mathcal{Q}$ of queries. For each query q a document set $\mathcal{D}_q$ must be ranked. Each of the rankers produces its own ranked list for each query. The ranked lists are then combined using a multileaving algorithm to produce a single ranked list. This multileaved list is then presented to a user, whose clicks are recorded. The multileaving method must infer the quality of the individual rankers based on the user's clicks. A multileaving method therefore has two distinct phases, a document sampling stage, where the multileaved list is created, and a ranker scoring phase, where each ranker is given some score based on the user clicks.

Recall from Section 2.3.3 that A/B testing can be regarded as the gold standard of online evaluation. We therefore regard the quality of a multileaving method as measuring its degree of agreement with A/B testing. Ranker evaluation based on A/B testing involves randomly assigning users to different rankers, and then for each ranker comparing the mean for some metric across user impressions. A ranker has then outperformed another ranker if its mean for the metric is greater than that of the other ranker. Given a set of rankers and a metric, the expected scores for each ranker for the metric on an A/B test induce an ordering on the rankers. We regard the ground truth to be this ordering. Our primary error metric for a multileaving method will be the percentage of pairwise errors in the preferences inferred by the multileaving method relative to this ordering.

We are primarily interested in three aspects of the error metric: Firstly, what amount of error would a multileaving method converge to, given an unlimited number of user impressions. We will refer to this as the *accuracy* of the multileaving method. Secondly, how quickly does the multileaving method converge to varying levels of accuracy, i.e. how many user impressions are needed to get to an error rate of 25% or 10%. We will refer to this as the *efficiency* of the multileaving method. Finally, we are interested in how these first two aspects are affected by the number of rankers being

simultaneously compared by the multileaving method. We will refer to this as the *scalability* of the method.

Note that for our experiments we do not have access to real users on which we can carry out A/B tests. Instead of comparing the inferred preferences of the multileaving against the ordering induced by their scores on A/B tests, we will therefore use the ordering induced by an offline metric as our ground truth. Unless otherwise stated, this metric will be the NDCG@10 score of the rankers. This only requires relevance judgements in order to compute the NDCG@10 score, and these are provided for many publicly available learning to rank datasets.

Bias has received a lot of attention in the interleaving and multileaving literature. A method is defined to be biased if it infers preferences between rankers for a user whose clicks are uniformly random between documents [56]. Bias is problematic since it should be impossible to determine which ranker is better if clicks are completely random. Unfortunately, this definition of bias does not capture susceptibility to a user's position bias. If a user exhibits position bias, previous multileaving algorithms can infer a ranker preference even if the user has no such preference. Since position bias is an extremely important source of bias in user behaviour, it would be preferable to define a method as being biased if it infers preferences between rankers for a user whose clicks are random, but display position bias. We will refer to this as the *latent bias* of a multileaving method and investigate how this affects the multileaving methods in Section 3.3.3.2.4. We call it latent bias to distinguish from the definition of bias commonly used for statistical estimators [121].

Since our ultimate goal is to minimize pairwise errors between the inferred preferences of a multileaving method and the ordering induced by an A/B test, it is not sufficient to create a method which does not display latent bias. Instead, we wish to ensure that the expected inferred preferences of the multileaving method agree with the expected ordering of the A/B test. We will see in Section 3.3 that this can be guaranteed by our second proposed method MIS.

A final property we are interested in is the possibility of the outcomes of multileaved comparisons to be distorted by the makeup of the comparison set. In the ideal case, when we carry out A/B tests to evaluate a set of rankers, there is a single best ranker. For interleaving and multileaving methods, this is not necessarily the case. For example, an interleaving or multileaving method might find that a ranker A is superior to a ranker B, which in turn is superior to a ranker C, but that ranker C is superior to A [42].

For multileaving methods the even more counterintuitive possibility can arise that even though there may be a single best ranker in a set of rankers when the rankers are compared pairwise, a different ranker may appear to be the best when three rankers are compared at a time. For example a ranker A could be superior to rankers B and C in pairwise comparisons, but when all three rankers are multileaved, B is superior to A and C, because the documents sampled into the multileaving systematically favour ranker B. Simple examples of how this can occur are given for TDM, PM and SOSM in Section 2.3.5. This problem of *distortion* was discovered for SOSM [18], and we will investigate how it affects different multileaving methods in Section 3.3.3.2.5.

3.2 Sample-Only Scored Multileaving

Recall from Section 2.3.5 that TDM does not scale well with the number of rankers being compared, since a click can only be used to infer something about the quality of a single ranker. Conversely, PM has the potential to be more efficient, since it infers something about the quality of each ranker based on each click. Unfortunately as noted in Section 2.3.5 PM can be highly inaccurate and we will see that it suffers from potentially extreme latent bias in Section 3.2.2.

We now introduce a multileaving method, Sample-Only Scored Multileaving (SOSM), which is efficient and scales well with the number of rankers being compared, since clicks are used to infer something about the quality of each ranker. SOSM is more efficient than, and similarly accurate to, TDM and substantially more accurate than PM.

3.2.1 Algorithm

Our aim is to create an algorithm which uses click feedback efficiently, unlike TDM, but does not introduce latent bias like PM. The core concept underlying SOSM is that even though the document sampling stage of SOSM may favour sampling documents from certain rankers, the scoring function will compensate for this imbalance by only scoring a ranker in terms of its relative ranking of the documents included in the multileaving.

The process of creating the multileaved list in SOSM is identical to that in TDM. SOSM creates the multileaved list in rounds. In each round a random ordering of the rankers is decided and the top document from each ranker that has not yet appeared in the multileaving is drawn until the multileaved list is of sufficient length.

To infer preferences, each ranker r first ranks the documents in the multileaved list such that $pos_M(d, r)$ denotes the order of document d among the documents in the multileaved list, according to their positions in the original ranked list of r.

Letting $\mathcal{D}_M$ denote the documents of the multileaved list, the score of document d for ranker r is given in Equation 3.1

$$s(pos_M(d, r)) = \frac{\frac{1}{pos_M(d,r)^3}}{\sum_{d' \in \mathcal{D}_M} \frac{1}{pos_M(d',r)^3}} \tag{3.1}$$

Finally, ranker r is credited with a final score of

$$f(r) = \sum_{d \in C} s(pos_M(d, r)),$$

where $\mathcal{C}$ denotes the set of clicked documents. Note the similarity between our scoring function in Equation 3.1, and the scoring function used by PM [98]. The only difference is that PM considers the positions of documents according to the original ranked lists when scoring rankers, whereas we only consider the relative positions of the documents in the multileaving according to the original ranked lists.

SOSM is efficient, accurate, and scales well with the number of rankers being compared, as verified experimentally in Section 3.2.2. However unfortunately SOSM displays latent bias. Note that SOSM does not suffer from

TABLE 3.1: Datasets. Each dataset consists of a number of query-document pairs, together with a relevance judgement for the pair. Each document is represented by a feature vector.

Datasets	Queries	URLs	Features
MSLR-WEB30K [1]	31,531	3,771,125	136
YLR Set 1 [25]	19,944	473,134	700
YLR Set 2 [25]	1,266	34,815	700

latent bias according to the definition used in previous interleaving and multileaving papers [56], but only according to our extended definition of latent bias.

3.2.2 Experimental Evaluation

We compare our algorithm, SOSM, against TDM and PM, the two previously state of the art multileaving methods. We begin by describing our experimental setup in Section 3.2.2.1, before presenting our results in Section 3.2.2.2.

3.2.2.1 Experimental Setup

Since we do not have access to real users for our experiments, we simulate the interactions for our multileaving methods using probabilistic user models. This experimental setup using simulated user interactions has become the standard for evaluating multileaving methods, and has been used to evaluate all the prior work in multileaving [102, 98, 17].

We use standard online learning to rank datasets, whose properties are summarised in Table 3.1. Following [102, 98, 17], for each dataset we choose the possible rankers to be the features of the dataset. That is, for a given feature, we construct a ranker which ranks documents only according to the score of that feature. An example of a feature in the MSLR dataset is the BM25 score of the body of the document. Although these rankers are weaker than the rankers that need to be evaluated in a practical use case, this is the standard evaluation methodology in the interleaving and multileaving literature. Using these feature rankers makes experimental replication easier. Furthermore, when investigating evaluation methods, it is the relative quality of the rankers, rather than the absolute quality of the rankers which determines how difficult the evaluation task is, and many of the feature rankers are very similar in quality.

For each experimental run we randomly sample K rankers, fix the number of iterations for the experiment to run, and choose a user model. We split the queries of the dataset into a test set and training set, and for each iteration we randomly sample a query with replacement from the training set, and each of the K rankers produces a ranked list. Each multileaving method then uses these ranked lists to produce a multileaved list, of length fixed to 10, which is displayed to a simulated user. The simulated user then clicks on some of the

TABLE 3.2: User Models. For each user model, the user inspects the ranked list from top to bottom and either clicks on a document, or stops inspecting the list, with each action's probability defined by the user model, and the document relevance. We use the perfect, navigational and informational click models from [56], additionally we define two random click models for testing bias.

	Click Probabilities					Stop Probabilities				
Relevance	0	1	2	3	4	0	1	2	3	4
Perfect	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
Navigational	0.05	0.1	0.2	0.4	0.8	0.0	0.2	0.4	0.6	0.8
Informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5
Random	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.0	0.0	0.0
Random Position Bias	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

documents according to the chosen cascade user model [56]. The parameters of these user models are described in Table 3.2. The perfect, navigational, and informational click models introduce progressively more noise into the user's interactions with the search system. The random click models are included to detect latent bias in the multileaving methods. Specifically, if a multileaving method detects a preference for one ranker over another when clicks are uncorrelated with document relevance, the multileaving method has displayed latent bias towards that ranker.

Based on the cumulative scores for two rankers i and j: $f_t(i)$ and $f_t(j)$ after t iterations according to the multileaving methods, we can create a matrix $\hat{M}(t)$ with entries $\hat{M}_{ij}(t) = \frac{f_t(i)}{f_t(i)+f_t(j)}$. We let a matrix P be given with entries $P_{ij} = \frac{g_i}{g_i+g_j}$, where g_i is the ground truth score of ranker i on a held-out test set of queries. Unless otherwise stated, this will be the NDCG@10 score of the ranker.

For a given pair of rankers, we consider the multileaving method to have made an error after t comparisons if $\hat{M}_{ij}(t)$ and P_{ij} disagree. We wish to minimize the percentage of errors made,

$$E(t) = \frac{\sum_{i,j \in R} sgn(\hat{M}_{ij}(t) - 0.5) \neq sgn(P_{ij} - 0.5)}{|R|(|R|-1)} \tag{3.2}$$

For the experiments involving random click models, instead of considering the error measure from Equation 3.2 as we do for all other experiments, we consider the multileaving method to have made an error if it detects a score ratio of 53% or more between a pair of rankers, since if the method was unbiased, this ratio would be expected to be 50%.

For the experiments not involving a random click model, we include a curve showing how well the NDCG score on the queries seen by the multileaving methods agrees with the NDCG score on the test set used to define the ground truth. This oracle, whose curve is labelled as NDCG in our figures, serves to establish a lower bound on the error that can reasonably be obtained. Unlike the multileaving methods, which only have access to the simulated user's implicit feedback, the oracle has access to the actual relevance judgements for the documents.

3.2.2.2 Experimental Results

Table 3.3 enumerates the percentage error after 2,000 and 10,000 iterations when multileaving 5, 40 or 100 rankers for the three click models. In almost all cases SOSM is superior. The two exceptions occur when comparing only 5 rankers. In this case, TDM is marginally better than SOSM for the informational click model, and indistinguishable for the perfect click model. For 40 or 100 rankers, SOSM substantially outperforms TDM and PM. For example, with 100 rankers and the informational click model, the error rates are reduced by 50% from 32% to 16% after 10,000 iterations.

TABLE 3.3: Percentage error, E, after 2,000 and 10,0000 iterations for each multileaving method for 5, 40 and 100 rankers and three click models. The best performing method is indicated with a *. Bolded entries indicate that SOSM is significantly different to the two baselines with $p < 0.01$.

	Iterations	2,000			10,000		
Click Model	Method / # Rankers	TDM	PM	SOSM	TDM	PM	SOSM
Perfect	5 rankers	18%	23%	18%*	14%	18%	14%*
	40 rankers	23%	28%	**17%***	18%	28%	**15%***
	100 rankers	30%	29%	**18%***	21%	28%	**16%***
Navigational	5 rankers	27%	30%	22%*	24%	25%	16%*
	40 rankers	34%	30%	**24%***	26%	31%	**22%***
	100 rankers	39%	31%	**25%***	29%	29%	**21%***
Informational	5 rankers	18%*	29%	20%	16%*	32%	18%
	40 rankers	37%	32%	**22%***	27%	32%	**15%***
	100 rankers	42%	33%	**21%***	34%	32%	**16%***

Figure 3.1 shows how the performances of the multileaving methods are affected by the click model used. For all three click models, SOSM outperforms both TDM and PM. This is most pronounced for the navigational model, where the percentage error for TDM and PM is 50% greater than that of SOSM (30% compared to 20%).

Figure 3.2 shows the sensitivity of the multileaving methods to different choices of dataset. We repeat the experiment from Figure 3.1(b) using two other datasets. All three algorithms (TDM, PM, SOSM) perform similarly across datasets. In all cases, SOSM exhibits superior performance.

Figure 3.3 shows how the performances of the multileaving methods scales with the number of rankers being compared. We show the percentage error after 2,000 iterations, as a function of the number k of rankers that are multileaved. For a given k, a random subset of rankers is selected and multileaved for 2,000 iterations. The same subset is used for PM, TDM and SOSM. This is repeated 25 times, each time with a different random subset of k rankers. The results in Figure 3.3 are the average of these 25 runs. We observe that for SOSM and PM the error remains relatively stable as the number of rankers increases. However, for TDM the error is increasing and we observe that its performance becomes worse than PM for large numbers of rankers. This is due to the fact that during the scoring phase, TDM is unable to assign credit to more than the 10 rankers from which the multileaved documents originated.

Figure 3.4 tests if the multileaving methods suffer from latent bias. In Section 2.3.5.1, we showed that PM could exhibit latent bias under certain conditions. For this experiment we use a random click model, i.e. clicks are random and independent of document relevance. In this case, we expect that the elements of the pairwise preference matrix should converge to 0.5, i.e. there is no observed preference between rankers i and j. An error is declared if the value of $\hat{M}_{ij}(t)$ deviates from 0.5 by more than 0.03. Figure 3.4 shows that PM exhibits very strong latent bias, with error rates of about 60%.[2] TDM's behaviour is much better, but after 2000 iterations some latent bias is still present. In contrast, the percentage error decreases much quicker in SOSM, and is almost zero after just 2000 iterations. Note that we will see in Section 3.3.3.2.4 that SOSM can suffer from latent bias when the user displays position bias in clicking behaviour.

3.2.3 Conclusions

We identified and experimentally verified weaknesses in the scalability of TDM and the unbiasedness of PM. We then proposed a new algorithm, SOSM, that corrects these problems. Experimental results using simulated users (perfect, navigational, informational click models), on three different datasets confirmed that (i) SOSM scales well with the number of rankers to be multileaved, (ii) is unbiased, and (iii) is significantly more efficient and accurate than prior methods. In some cases error rates were reduced by half.

The residual error needs investigating but is likely to be partly due to (i) establishing a ground truth based on NDCG@10, which is not used as the scoring function in Equation 3.1, and (ii) the ground truth data is computed on "test" data that is not used during the multileaving experiments.

Although SOSM is highly efficient, the scoring function is based only on the documents included in the multileaving. This can make it difficult to accurately capture certain ground truths, since for example, it would be difficult to use SOSM to obtain good agreement with a ground truth which only credits the top ranked documents of rankers. We now introduce a multileaving method with a more adaptable method for scoring rankers, and for which we can provide guarantees on its accuracy.

3.3 Multileaving Using Importance Sampling

We will first motivate Multileaving using Importance Sampling (MIS) and show that we can make the expected outcome of a multileaving equal to the expected outcome under A/B testing in Section 3.3.1. We will then describe the MIS algorithm in Section 3.3.2. We will experimentally evaluate MIS in Section 3.3.3. Finally, we will conclude our multileaving chapter in Section 3.3.4

3.3.1 Motivation

Recall from Section 3.1 that our goal is to create a multileaving method which produces the same relative outcomes between rankers as A/B testing

[2] Note that in [98] no such bias was detected. However, in [98] the set of multileaved rankers was not picked randomly. Further, personal communication with an author of [98] confirmed the existence of a software bug in PM which we corrected for these experiments.

for a given metric. One way of guaranteeing this is to create a multileaving method which ensures that the expected score for each ranker is equal to the expected score of that ranker under A/B testing with a given metric. This is possible using importance sampling, and we will now demonstrate how it can be done.

For the following, let the metric use some click scoring function s to evaluate the rankers. The scoring function credits rankers based on the rank position of documents' users click. We restrict our attention to functions that only credit documents in the top-10 of a ranker, i.e.

$$s(i) = \begin{cases} t(i) & i \in \{1, \dots 10\} \\ 0 & \text{else} \end{cases}$$

where examples of t include:

$$t(i) = 1/i$$
$$t(i) = 1/i^2$$
$$t(i) = 1/\log(i+1)$$

We will begin by considering the expected outcome if we were carrying out A/B testing. We assume that we are given a set of $\mathcal{R}$ rankers and a set $\mathcal{D}$ of documents. Each ranker displays a ranked list of the documents in $\mathcal{D}$ to a user, whose clicks are recorded. If we denote by $\mathcal{C}$ the set of clicked documents, and $pos(d, R)$ the position of document d according to ranker R, then the final score f for each ranker $R \in \mathcal{R}$ is given by

$$f(R) = \sum_{d \in \mathcal{C}} s(pos(d, R)), \tag{3.3}$$

Let us assume that the probability of a document being clicked on, i.e. its click probability, is independent of its position (beyond needing to be in the top-10), and independent of what other documents are present in the ranked list. Let $p(c_d|d \in T^{10})$ denote document d's click probability, given that it was included in the top-10. Note that $p(c_d|d \notin T^{10}) = 0$, since we assume that users do not see or click on documents outside the top-10. Let $p_R(d \in T^{10})$ denote the probability that d is included in the top-10 of ranker R. Then, by definition,

$$p_R(d \in T^{10}) = \begin{cases} 1 & \text{if } d \in T^{10} \\ 0 & \text{else} \end{cases}$$

and we can write the probability of a click on document d given that ranker R has been displayed to the user as

$$p_R(c_d) = p(c_d|d \in T^{10}) p_R(d \in T^{10}) \tag{3.4}$$

We can write the expected value of $f(R)$ under the A/B testing regime as

$$\mathbb{E}[f(r)] = \sum_{d \in \mathcal{D}} p_R(c_d) s(pos(d, R)) \tag{3.5}$$

$$= \sum_{d \in \mathcal{D}} p(c_d | d \in T^{10}) p_R(d \in T^{10}) s(pos(d, R)) \tag{3.6}$$

This is the expected value that we would like to obtain when multileaving.

When multileaving, we have a fixed probability for each document that it will be included in the top-10 of the multileaving displayed to the user, which we denote by $p_M(d \in T^{10})$. We then have that the probability of a click on document d, when multileaving, denoted by $p_M(c_d)$ is given by

$$p_M(c_d) = p(c_d | d \in T^{10}) p_M(d \in T^{10}) \tag{3.7}$$

If we were to record the empirical score f as in Equation 3.3, we would not obtain the same expected score when multileaving, since in general $p_M(d \in T^{10}) \neq p_R(d \in T^{10})$. Instead, it is possible to obtain the same expected score as under A/B testing by recording the empirical score g shown below instead of using f.

$$g(R) = \sum_{d \in \mathcal{C}} \frac{s(pos(d, R))}{p_M(d \in T^{10})} \tag{3.8}$$

The idea is to use importance sampling to correct for the modified document sampling probabilities when multileaving. We require that $p_M(d \in T^{10})$ is known, which is the case since we control the document sampling stage of the multileaving. Furthermore we require the document sampling stage to satisfy that $p_M(d \in T^{10}) \neq 0$ for all d such that there exists a ranker R in the comparison set such that $p_R(d \in T^{10}) \neq 0$. That is, any document which is present in the top-10 of at least one ranker must have a non-zero probability of being in the multileaving.

Then the empirical score has expected value

$$\mathbf{E}[g(R)] = \sum_{d \in \mathcal{D}_M} p_M(c_d) \frac{s(pos(d, R))}{p_M(d \in T^{10})} \tag{3.9}$$

$$= \sum_{d \in \mathcal{D}_M} p(c_d | d \in T^{10}) p_M(d \in T^{10}) \frac{s(pos(d, R))}{p_M(d \in T^{10})} \tag{3.10}$$

$$= \sum_{d \in \mathcal{D}} p(c_d | d \in T^{10}) s(pos(d, R)) \tag{3.11}$$

$$= \sum_{d \in \mathcal{D}} p(c_d | d \in T^{10}) p_R(d \in T^{10}) s(pos(d, R)) \tag{3.12}$$

$$= \mathbf{E}[f(R)] \tag{3.13}$$

where we have used that $p_R(d \in T^{10}) = 1$ if $s(pos(d, R)) > 0$, and $p_R(d \in T^{10}) = 0$ otherwise. We have now shown how to create a multileaving algorithm for which the expected score for each ranker is equal to the expected score of the ranker under A/B testing. In Section 3.3.2 we describe our implementation of this idea.

3.3.2 Algorithm

Recall that a multileaving algorithm has a document sampling stage and a ranker scoring stage. We describe these next for our algorithm.

3.3.2.1 Document sampling stage

As noted in Section 3.1, we require that for any document which appears in the top-10 of some ranker, this document has a non-zero probability of appearing in the top-10 of the multileaving. Apart from this requirement, the document sampling strategy can be chosen freely according to any number of criteria. We choose the following document sampling strategy which makes the importance weights from Equation 3.16 easy to compute.

Given a set $\mathcal{R}$ of rankers, and a set $\mathcal{D}_q$ of documents to be ranked for a given query, we first filter out all the documents which are not included in the top-10 of at least one of the rankers. We denote by N_q the number of documents remaining after the filtering step. We then rank all the remaining N_q documents according to an arbitrary preference criterion, such as, for instance, the simple or weighted average rank of the document across all rankers. We then partition the document set into a set of M 'preferred' documents and $N_q - M$ 'non-preferred' documents, where the preferred documents are the top M documents according to the chosen preference criterion. We can then sample a proportion $L \in [0, 1]$ of the M preferred documents uniformly at random, and sample the remaining documents needed to obtain a multileaving of the desired length from the non-preferred documents uniformly at random. Finally, we display a random ordering of the sampled documents to the user. The purpose of the parameter L is to control how heavily we favour sampling from the preferred documents.

The pseudocode of the above is displayed in Algorithm 1.

For a fixed multileaving of length 10, it is simple to verify that

$$p_M(d \in T^{10}) = \frac{10L}{M} \tag{3.14}$$

for preferred documents, and

$$p_M(d \in T^{10}) = \frac{10(1-L)}{N_q - M} \tag{3.15}$$

for non-preferred documents.

3.3.2.2 Ranker Scoring Stage

Recall from Section 3.1 that we have restricted our attention to scoring functions which only credit documents in the top-10 of a ranker. More specifically, we will credit a ranker R with $s(pos(d, R))$ for each clicked document d, where

$$s(pos(d, R)) = 1/(log(1 + pos(d, R)))$$

if $pos(d, R) \leq 10$, and $s(pos(d, R)) = 0$ otherwise. We choose a logarithmic decay in the scoring function to match the logarithmic decay in the NDCG ground truth (which is the most commonly used evaluation metric in the multileaving literature). As seen in Section 3.1, we need to normalise each

document's credit by its probability of being included in the top-10 of the multileaving to obtain our final ranker score. We do this as follows:

$$g(R) = \sum_{d \in \mathcal{C}} \frac{s(pos(d, R))}{p_M(d \in T^{10})}, \tag{3.16}$$

where $p_M(d \in T^{10})$ is the probability of that document appearing in the top-10 of the multileaving and $\mathcal{C}$ denotes the set of clicked documents. The need to compute $p_M(d \in T^{10})$ for each document is the reason we chose our document sampling strategy, since for our strategy the probabilities are simple to compute, as shown in Equations 3.14 and 3.15.

Parameters: M, L
$S_R = 0$ for all rankers
for $q = 1, \ldots, T$ **do**
  Filter out any documents not ranked in the top-10 of some ranker
  Rank the remaining documents according to average rank across all rankers
  Let the top-M of these documents according to this ranking be 'preferred' documents, and the remaining documents be 'non-preferred'
  Sample L of the M 'preferred' documents and the remaining documents from the 'non-preferred' set
  Display sampled documents in random order to user and observe clicks
  Let C_q be the set of documents that were clicked on
  for $R = 1, \ldots, K$ **do**
    $g(R) \leftarrow g(R) + \sum_{d \in C_q} [s(pos(d, R))/p_M(d \in T^{10})]$
  end
end

Algorithm 1: Multileaving Using Importance Sampling (MIS)

We refer to our multileaving method that uses the document sampling stage and ranker scoring stage presented above as *Multileaving Using Importance Sampling* (MIS).

3.3.3 Experimental Evaluation

We compare our method, MIS, against SOSM [17] and TDM [102]. We omit PM from the experiments because its performance was shown to be substantially worse than TDM and SOSM in [17], see Section 3.2.2.2 for details.

3.3.3.1 Experimental Setup

We replicate the exact same experimental setup of [102, 98, 17], where user interactions are simulated using probabilistic user models. We use four standard online learning to rank datasets, whose properties are summarised in Table 3.4. Following [102, 98, 17], for each dataset we treat the features of the dataset as rankers. That is, for a given feature, we construct a ranker which ranks documents only according to the score of that feature.

TABLE 3.4: Datasets. Each dataset consists of a number of query-document pairs, together with a relevance judgement for the pair. Each document is represented by a feature vector.

Datasets	Queries	URLs	Features
MSLR-WEB30K [3]	31,531	3,771,125	136
YLR Set 1 [25]	19,944	473,134	700
YLR Set 2 [25]	1,266	34,815	700
Yandex [4]	9,124	97,290	245

For each run, we randomly sample K rankers, fix the number of iterations, and choose one of the user models displayed in Table 3.5. We split the queries and associated relevance judgements of the dataset into two subsets A and B[5]. For each iteration we randomly sample a query with replacement from the set A, and each of the K rankers produces a ranked list. Each multileaving method then uses these ranked lists to produce a multileaved list, of length fixed to 10, which is displayed to a simulated user. The simulated user then clicks on some of the documents according to the chosen user model [56].

In these user models, the simulated user inspects the displayed list from the top, inspecting one document at a time. After inspecting each document, there is a probability that the user clicks on a document, or stops browsing. This probability is controlled by parameters described in Table 3.5. The perfect, navigational, and informational click models introduce progressively more noise into the user's interactions with the search system, and reflect different types of user behavior. The perfect click model represents a user whose click probabilities are perfectly correlated with document relevance. The navigational click model represents a user looking for a specific result. This click model therefore has a high probability of clicking on highly relevant documents, and a high probability of stopping browsing after encountering highly relevant documents. The informational click model represents a user searching for general information about a topic. Taken together, these user models represent some of the most common types of user behavior.

Each multileaving method maintains a matrix $\hat{M}(t)$ with entry $\hat{M}_{ij}(t)$ storing the score ratio of rankers i and j after t iterations. Additionally, we let a matrix P be given with entries $P_{ij} = \frac{g_i}{g_i+g_j}$, where g_i is the ground truth NDCG@10 score of ranker i on the held-out set B of queries. This set is used only for computing the ground truth scores of the rankers. Then, for a given pair of rankers, we consider the multileaving method to have made an error after t comparisons if $\hat{M}_{ij}(t)$ and P_{ij} disagree. We wish to minimize

[5]This corresponds to the split into training and test sets already contained in the datasets. We follow the convention from [102, 98, 17] of splitting the datasets, but prefer not to use the words training and test, since there is no explicit training.

TABLE 3.5: User Models. For each user model, the user inspects the ranked list from top to bottom and either clicks on a document, or stops inspecting the list, with each action's probability defined by the user model, and the document relevance. We use the perfect, navigational and informational click models from [56], additionally we define two random click models for testing latent bias.

	Click Probabilities					Stop Probabilities				
Relevance	0	1	2	3	4	0	1	2	3	4
Perfect	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
Navigational	0.05	0.1	0.2	0.4	0.8	0.0	0.2	0.4	0.6	0.8
Informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5
Random	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.0	0.0	0.0
Random Position Bias	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

the percentage of errors made,

$$E(t) = \frac{\sum_{i,j \in R} sgn(\hat{M}_{ij}(t) - 0.5) \neq sgn(P_{ij} - 0.5)}{|R|(|R| - 1)} \tag{3.17}$$

For the experiments not involving a random click model, we include a curve showing how well the NDCG score on the queries seen by the multileaving methods agrees with the NDCG score on the test set used to define the ground truth. This oracle, labelled as NDCG in our figures, serves to establish a lower bound on the error that can reasonably be obtained. Unlike the multileaving methods, which only have access to the simulated user's implicit feedback, the oracle has access to the actual relevance judgements for the documents.

3.3.3.2 Experimental Results

We are primarily interested in the accuracy, efficiency and scalability of multileaving methods.

We evaluate the methods in terms of our three main criteria: accuracy in Section 3.3.3.2.1, efficiency in Section 3.3.3.2.2 and scalability in Section 3.3.3.2.3. Finally, we investigate how latent bias affects the methods in Section 3.3.3.2.4 and to what extent the methods are affected by distortion in Section 3.3.3.2.5. Additionally, we investigate the impact of the parameter settings on the performance of Multileaving using Importance Sampling in Section 3.3.3.2.6.

3.3.3.2.1 Accuracy

Figure 3.5 shows the performance of the multileaving algorithms on the MSLR dataset for the perfect, navigational and informational click models. We are interested in what percentage of errors the methods converge to. We include a curve showing how well the NDCG score computed on the queries seen by the multileaving methods agrees with the NDCG ground truth. This curve, labelled as oracle in our figures, serves to establish a lower bound on the error that can reasonably be obtained. Unlike the multileaving methods,

which only have access to the simulated user's implicit feedback, the oracle has access to the actual relevance judgements for the documents.

In the legends of our figures we note the average NDCG@10 scores for the multileaved lists displayed by the different multileaving methods. This is the mean across the experimental runs of the NDCG of the displayed multileaved lists.

We see in Figure 3.5 that MIS is substantially more accurate than both TDM and SOSM for all click models. For example for the perfect click model after 100,000 iterations, the error rate for TDM and SOSM are 15.5% and 16.0% respectively. In comparison, for $M = 0$, the error rate for MIS was only 3.3%, an improvement by a factor of approximately 5 over TDM and SOSM. For $M = 10, L = 0.8$, the error rate for MIS was 7.1%, an improvement by a factor of more than 2 over TDM and SOSM.

Furthermore, unlike TDM and SOSM, which appear to have stopped improving after 100,000 iterations, the curves have not fully levelled off for MIS. It is therefore likely that these curves present an underestimate of the accuracy gain from MIS relative to TDM and SOSM. Indeed, for additional experiments which ran for 200,000 iterations on the MSLR dataset with the perfect click model, the mean percentage error decreased from 7.1% after 100,000 iterations to 6.3% after 200,000 iterations for MIS with $M = 10, L = 0.8$. In comparison, for TDM the error had only decreased from 15.5% after 100,000 iterations to 15.4% after 200,000 iterations. For SOSM the error remained unchanged at 16.0% between 100,000 and 200,000 iterations.

Figure 3.6 shows the performance of the multileaving algorithms on the YLR1, YLR2 and Yandex datasets for the perfect click model. Again, MIS is substantially more accurate than TDM and SOSM for all datasets. Similar results were obtained for both navigational and informational click models but are omitted due to space constraints.

Tables 3.6 and 3.7 show the percentage error for each multileaving method after 2,000, 20,000, 50,000 and 100,000 iterations. Table 3.6 shows the performance for the perfect, navigational and informational click models for the MSLR dataset and Table 3.7 shows the performance for the perfect click model for the YLR1, YLR2 and Yandex datasets. We note that for all parameter settings, and across all the datasets and click models, MIS is substantially and significantly more accurate than TDM and SOSM.

3.3.3.2.2 Efficiency

Recall that the efficiency of a multileaving method measures how quickly the multileaving method converges to varying levels of accuracy, i.e. how many user impressions are needed to get to an error rate of 25% or 10%. The efficiency of MIS decreases as we increase the L parameter. Figure 3.5 shows that the efficiency of MIS is most impacted by the noise in the click model for greater values of L. Whereas the efficiency of MIS is good, when $M = 0$, and for $L = 0.6$, increasing L further to 0.8 results in the performance of MIS not overtaking SOSM until after approximately 40,000 iterations for the informational click model.

We note that the performance of MIS is less affected by the parameter settings, L and M, for the YLR1, YLR2 and Yandex datasets than for the MSLR dataset. This probably reflects the fact that the median size of the document pool for the MSLR dataset is much larger than those for the

TABLE 3.6: Percentage error, after 2,000, 20,000, 50,000 and 100,000 iterations for each multileaving method for 20 rankers and three click models on the MLSR dataset. The best performing method is indicated with a *. Bolded entries indicate that the method is significantly better than TDM and SOSM with $p < 0.01$ according to paired t-tests. Italicised entries indicate that the method is significantly worse than at least one of TDM and SOSM with $p < 0.01$.

		Method				
Click Model	Num. Iterations	TDM	SOSM	MIS (M=0)	MIS (M=10,L=0.6)	MIS (M=10,L=0.8)
Average NDCG		0.237	0.237	0.196	0.233	0.248
Perfect	2,000	19.7%	17.9%	**12.1%***	17.0%	*22.5%*
	20,000	16.3%	16.4%	**4.5%***	**7.6%**	**10.7%**
	50,000	15.7%	16.2%	**3.8%***	**6.8%**	**8.2%**
	100,000	15.5%	16.0%	**3.3%***	**6.1%**	**7.1%**
Navigational	2,000	30.9%	23.3%*	24.5%	*29.8%*	*34.5%*
	20,000	20.1%	19.4%	**12.7%***	**16.0%**	19.7%
	50,000	18.9%	18.9%	**10.4%***	**14.0%**	**15.9%**
	100,000	18.5%	18.9%	**9.7%***	**12.9%**	**14.4%**
Informational	2,000	31.3%	21.0%	*27.4%*	*32.5%*	*36.9%*
	20,000	18.9%	15.4%	**11.2%***	16.1%	*19.3%*
	50,000	16.3%	14.7%	**8.1%***	**11.7%**	**13.8%**
	100,000	15.5%	14.5%	**6.5%***	**9.4%**	**11.7%**

other datasets. For the MSLR dataset, the median number of documents to be ranked per query is 110, whereas it is 19 for the YLR1 and YLR2 datasets, and only 9 for the Yandex dataset. Since the number of documents determines how large the reweighting due to the document probabilities in the importance sampling step of MIS can be, see Equation 3.16, we expect that for a larger document pool, the reweighting will be a larger factor

In general, MIS is more efficient than TDM, requiring only 20,000 iterations to obtain a lower percentage error than TDM even in the worst case. MIS is also more efficient than SOSM at reaching high levels of accuracy, but reaches lower levels of accuracy slower than SOSM, in some cases requires as many as 40,000 iterations to outperform SOSM.

Note that while the number of iterations required for MIS to overtake SOSM may seem large, requiring up to 40,000 iterations in the worst case, this is a relatively small number of iterations compared to what is often required when A/B testing or interleaving in commercial search engines [26]. Here, it is not uncommon to require hundreds of thousands of iterations [26].

3.3.3.2.3 Scalability

Figure 3.7 shows how the performances of the algorithms scale with the number of rankers. We see that the only algorithm for which performance is adversely affected by the number of rankers being compared is TDM. This reflects the fact that unlike SOSM, and MIS, which gain information about the performance of each ranker at each iteration, TDM can only gain information about at most 10 rankers at each comparison.

3.3.3.2.4 Latent bias

Prior works on multileaving have measured whether multileaving algorithms have a bias towards a ranker when a user shows no such preference [102, 98,

TABLE 3.7: Percentage error, after 2,000, 20,000, 50,000 and 100,000 iterations for each multileaving method for 20 rankers, a perfect click model, and three data sets. The best performing method is indicated with a *. Bolded entries indicate that the method is significantly better than TDM and SOSM with $p < 0.01$ according to paired t-tests. Italicised entries indicate that the method is significantly worse than at least one of TDM and SOSM with $p < 0.01$.

		Method				
Dataset	Num. Iterations	TDM	SOSM	MIS (M=0)	MIS (M=10,L=0.6)	MIS (M=10,L=0.8)
YLR1	2,000	28.9%	21.6%	**20.5%***	22.1%	*25.6%*
	20,000	20.3%	14.6%	**12.9%***	**13.1%**	*16.1%*
	50,000	18.4%	13.5%	**10.8%***	**11.0%**	13.2%
	100,000	16.7%	13.0%	**10.2%**	**10.0%***	**11.8%**
YLR2	2,000	28.2%	23.3%*	23.4%	24.7%	*27.5%*
	20,000	22.2%	19.0%	**17.3%***	**17.4%**	19.5%
	50,000	20.5%	18.4%	**16.2%**	**15.8%***	**16.9%**
	100,000	19.8%	17.7%	**15.4%**	**14.8%***	**15.6%**
Yandex	2,000	29.2%	14.6%	14.1%	13.7%*	15.0%
	20,000	14.1%	7.2%	**6.4%***	**6.4%**	7.0%
	50,000	10.7%	6.2%	**5.2%**	**5.1%***	**5.4%**
	100,000	9.1%	5.8%	**4.5%**	**4.4%***	**4.7%**

17]. We refer to this as latent bias. To measure this, a random user click model is used, i.e. a user randomly clicks on documents with no preference for documents or rankers. In this case, the pairwise preferences derived from the score matrix $\hat{M}$, should *not* favour any ranker.

Figure 3.8 shows the performance of MIS against TDM and SOSM for random click models with varying degrees of position bias. The error here measures the extent to which preferences are detected even though under the random click models, there should not be any preferences. A failure to eventually reach a percentage error of 0 suggests latent bias in the method. We note that contrary to evaluations which only considered the random click model without position bias, Figure 3.8(b) shows that SOSM can be biased when the user displays position bias. These experiments detect no latent bias in TDM or MIS.

3.3.3.2.5 Distortions Analysis

We further analyse the possibility that the outcomes of multileaved comparisons may be distorted by the makeup of the comparison set. For multileaving methods the counterintuitive possibility can arise that even though there may be a single best ranker in a set of rankers when the rankers are compared pairwise, a different ranker may appear to be the best when three rankers are compared at a time. For example, ranker A could be superior to rankers B and C in pairwise comparisons, but when all three rankers are multileaved, B is superior to A and C, because the documents sampled into the multileaving systematically favour ranker B. Simple examples of how this can occur are given for TDM, PM and SOSM in Section 2.3.5. This problem of *distortion* was observed in [18], and we here investigate how it affects different multileaving methods.

Distortion can be quantified by first randomly sampling a fixed size subset of rankers that includes a ranker that beats every other ranker in the subset

in a pairwise comparison. This ranker is called the Condorcet winner. We then multileave all the rankers in the subset, and measure, after some fixed number of multileavings, the fraction of rankers that beat the Condorcet winner more than 50% of the time. If there is no distortion, and the number of multileavings is sufficient, we expect this fraction to be zero.

The following experiments test the level of distortion in the multileaving methods TDM, SOSM and MIS. For each dataset, and each click model, we randomly sample subsets of rankers of sizes 10 that include a Condorcet winner. We examine the probabilities of the rankers beating the Condorcet winner after 5,000 iterations. Note that this is likely to be an overestimate of the distortion of the multileaving method, since, for rankers of very similar quality, 5,000 iterations may not be sufficient to reliably distinguish rankers. Table 3.8 shows the average, over 100 runs, of the percentage of rankers that beat the Condorcet winner.

We observe that the distortion problem is largest for SOSM, and that it mostly manifests itself for the MSLR dataset with navigational and informational click models. This verifies the findings of [18]. There is evidence of distortion for TDM in the Yandex dataset. Almost no distortion is observed for MIS.

TABLE 3.8: Percentage of rankers beating the Condorcet winner (distortion), averaged over 100 runs, after 5,000 iterations for 20 rankers being multileaved for each dataset and click model.

	Distortion				
Method	TDM	SOSM	MIS (M=0)	MIS (M=10,L=0.6)	MIS (M=10,L=0.8)
MSLR Perfect	0.0%	0.0%	0.0%	0.0%	0.0%
MSLR Navigational	0.0%	1.3%	0.0%	0.0%	0.0%
MSLR Informational	0.0%	8.9%	0.0%	0.0%	0.0%
YLR1 Perfect	0.0%	0.0%	0.0%	0.0%	0.0%
YLR1 Navigational	0.0%	0.0%	0.0%	0.0%	0.0%
YLR1 Informational	0.0%	0.0%	0.0%	0.0%	0.0%
YLR2 Perfect	0.0%	0.0%	0.0%	0.0%	0.0%
YLR2 Navigational	0.0%	0.5%	0.0%	0.0%	0.0%
YLR2 Informational	0.0%	0.8%	0.0%	0.0%	0.0%
Yandex Perfect	0.0%	0.5%	0.0%	0.0%	0.0%
Yandex Navigational	1.8%	4.8%	0.0%	0.0%	0.3%
Yandex Informational	1.7%	5.1%	0.0%	0.0%	0.0%

3.3.3.2.6 Parameter Settings Analysis

Finally, we look into the impact of parameter L on the performance of our MIS method. Parameter L controls how much the sampling procedure favours the "preferred" documents.

Figure 3.9 shows that the efficiency of MIS gradually decreases as we increase the parameter L When L is high, we are selecting a larger fraction of "preferred" documents, thereby ensuring that the multileaved list is of high quality, i.e. has high NDGC@10. However, the variances of the scores increase as we increase L, due to the increased imbalances of the document probabilities (given by Equations 3.14 and 3.15), which therefore cause larger fluctuations in the reweighted document scores shown in Equation 3.16. This

TABLE 3.9: Mean and Variance of Scores for different parameter Settings. The last row shows the mean NDCG of the multileaved list displayed to users for the different parameter settings.

Parameter settings: M,L	$0, N/A$	$10, 0.2$	$10, 0.3$	$10, 0.4$	$10, 0.5$	$10, 0.6$	$10, 0.7$	$10, 0.8$	$10, 0.9$
Score Mean	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
Score Variance	2.28	2.28	2.28	2.45	2.91	3.59	4.40	6.11	11.65
NDCG@10	0.196	0.199	0.209	0.217	0.224	0.233	0.240	0.248	0.256

represents a form of exploration-exploitation tradeoff within the multileaving method. If L is low, we explore more by considering the documents more equally. Conversely, if L is high, we exploit our knowledge of which documents are likely to be good, at the expense of less information about the quality of low ranked documents, and high variance in score estimates.

These larger document score variances mean that the weighting of documents scores plays a larger role in the score differences of rankers, and the actual quality differences between rankers therefore requires more iterations, before their contributions to the scores begin to dominate. To illustrate the differences in variance we provide the score variances together with the score means for different parameter settings in Table 3.9. For smaller values of L, the variance of the scores is low, but it begins to substantially increase as L increases.

From Table 3.9 we also see the mean $NDCG$@10 scores of the multileaved lists displayed to the user for the different parameter settings. For comparison, the mean $NDCG$ of the multileaved lists displayed by TDM and SOSM is 0.237.

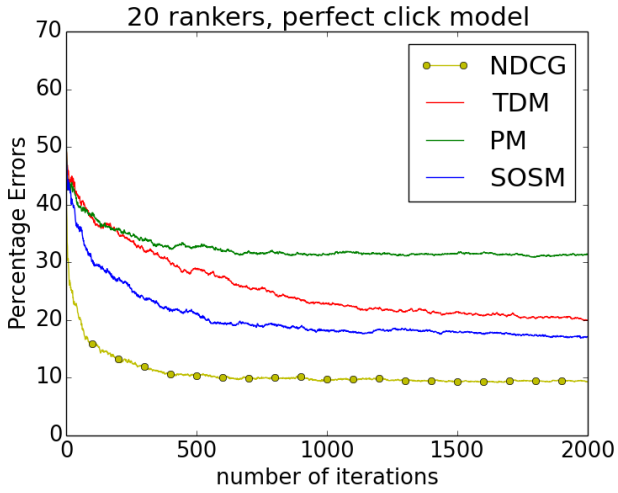
For all our experiments except those shown in Figure 3.9 and Table 3.9 we display the results for three parameter settings for MIS. This is to prevent the plots from being too difficult to read because of the number of curves included. We consider (i) M=0, i.e. full exploration, which maximises convergence at the expense of the quality of the multileaved list. Here the multileaved list has an average NDCG@10 of 0.196; (ii) M=10, L=0.6, which provides a multileaved list with average NDGC@10 of 0.233 which is slightly poorer that for TDM and SOSM; and (iii) M=10, L=0.8, which provides a multileaved list with average NDGC@10 of 0.248 which is slightly better than that for TDM and SOSM.

3.3.4 Conclusions

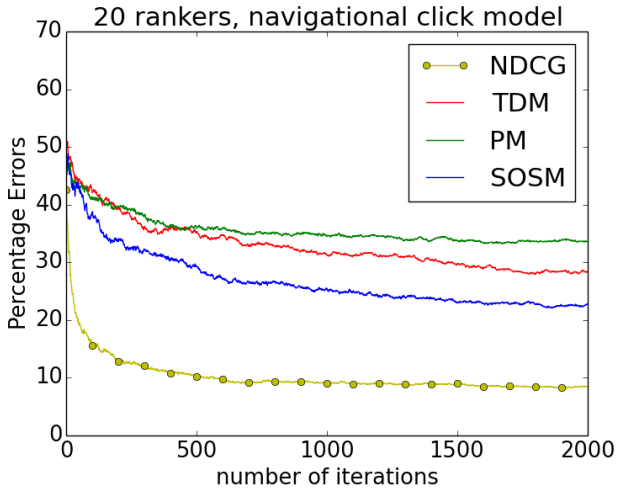
We have introduced a new multileaving method, MIS, for which we can guarantee that the expected outcome of comparisons agrees with A/B testing. Furthermore we have experimentally verified that this method is substantially more accurate than prior multileaving methods. For MIS there exists a tradeoff between the efficiency of learning, and the quality of the multileaved list, as controlled by the parameter L of the algorithm. MIS can be highly efficient if user feedback has little noise. Conversely, MIS can produce high quality ranked lists, but this comes at the expense of diminished efficiency, particularly for noisier user feedback.

In the future, we will investigate the possibility of using weighted importance sampling, and document sampling techniques which preferentially

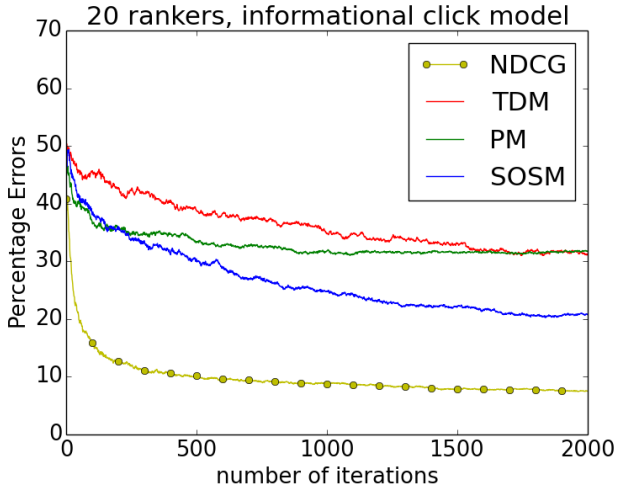
sample from those rankers for which our quality estimates are most uncertain, in order to improve the tradeoff between learning efficiency and quality of the multileaved list. Additionally, if this tradeoff can be handled better, there is the potential to investigate document sampling strategies for which we more strongly control the order of the documents presented during multileaving.



(A) perfect click model

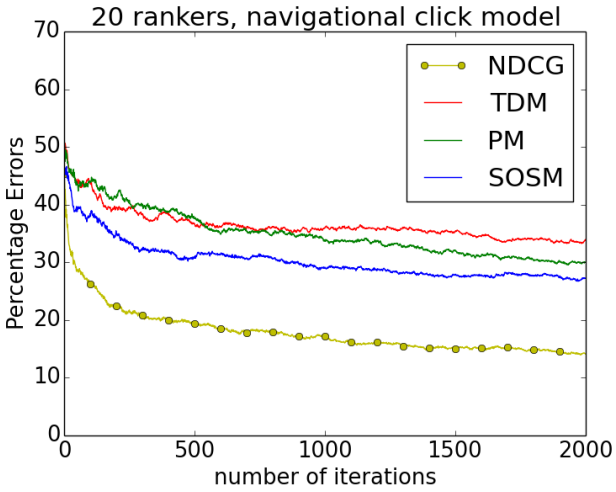


(B) navigational click model

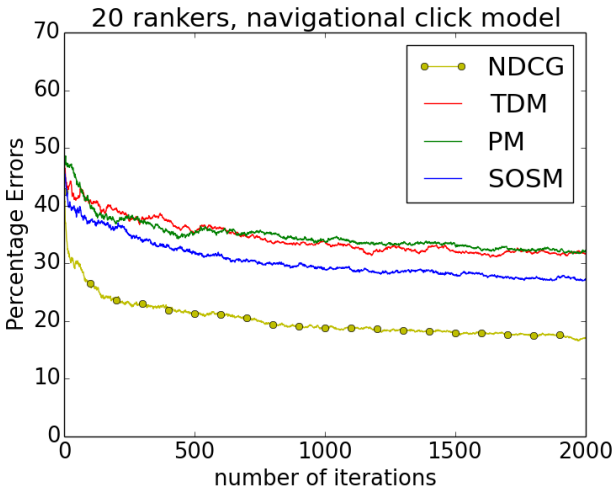


(C) informational click model

FIGURE 3.1: Percentage errors (averaged over 25 runs) versus the number of iterations on random subsets of 20 rankers for the MSLR dataset using perfect (a), navigational (b), and informational (c) click models.

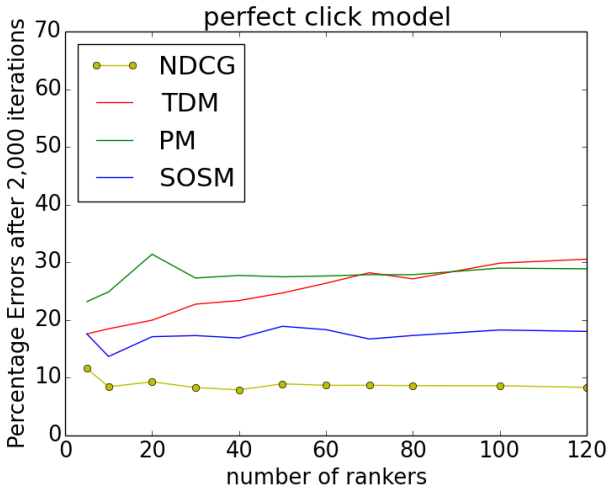


(A) YLR1

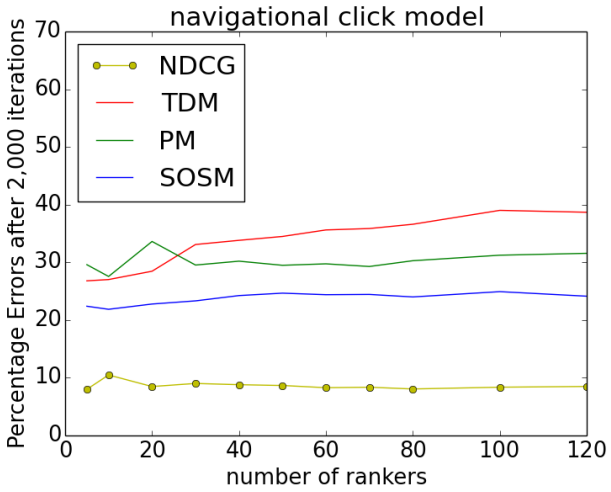


(B) YLR2

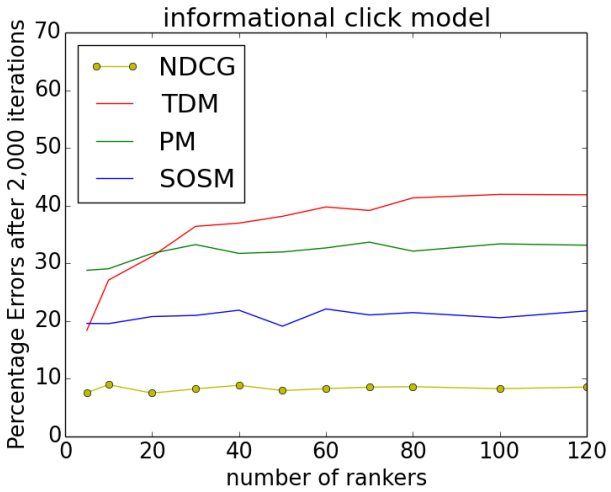
FIGURE 3.2: Percentage errors (averaged over 25 runs) versus the number of iterations on random subsets of 20 rankers for the YLR1 (a) and YLR2 (b) datasets using a navigational click model.



(A) perfect click model

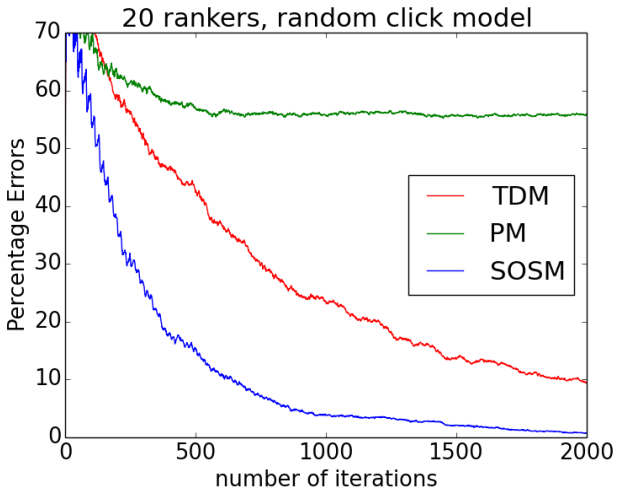


(B) navigational click model



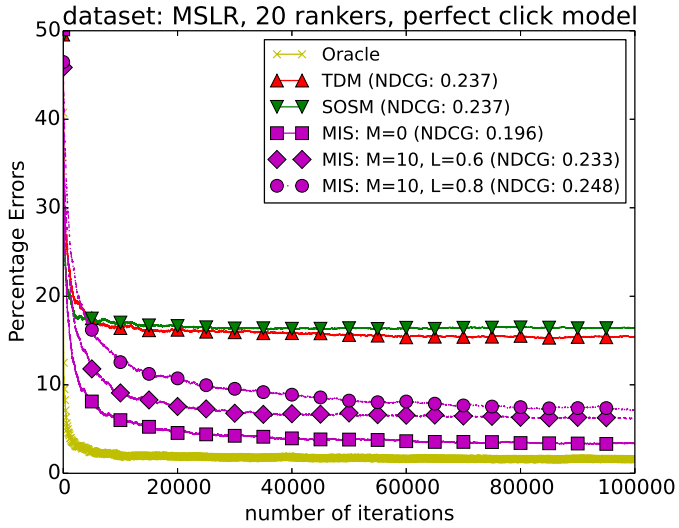
(C) informational click model

FIGURE 3.3: Percentage errors after 2,000 iterations (averaged over 25 runs) versus the number of rankers being compared on 25 random subsets of k rankers for the MSLR dataset with perfect (a), navigational (b), and informational (c) click models.

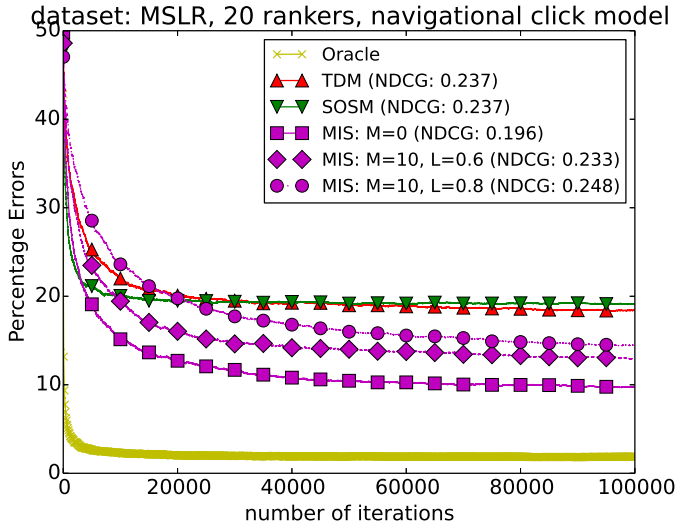


(A) random click model

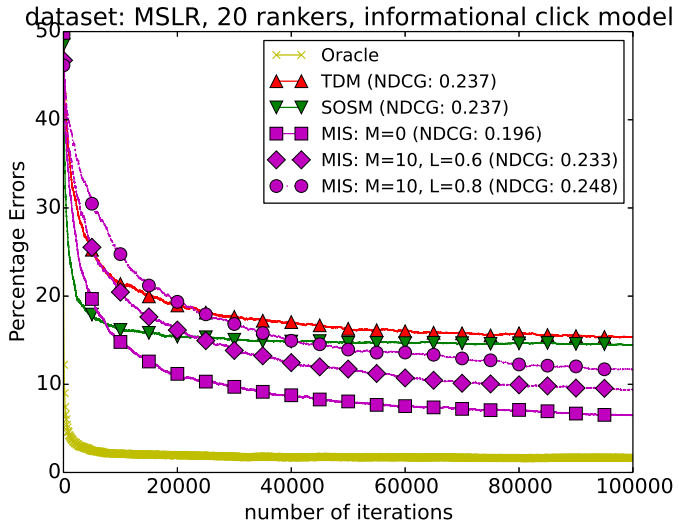
FIGURE 3.4: Percentage errors (averaged over 25 runs) versus the number of iterations on random subsets of 20 rankers for the MSLR dataset using a random click model.



(A)

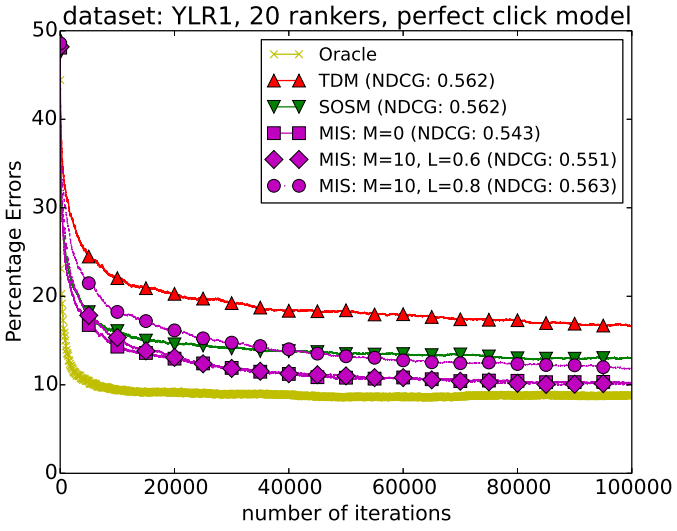


(B)

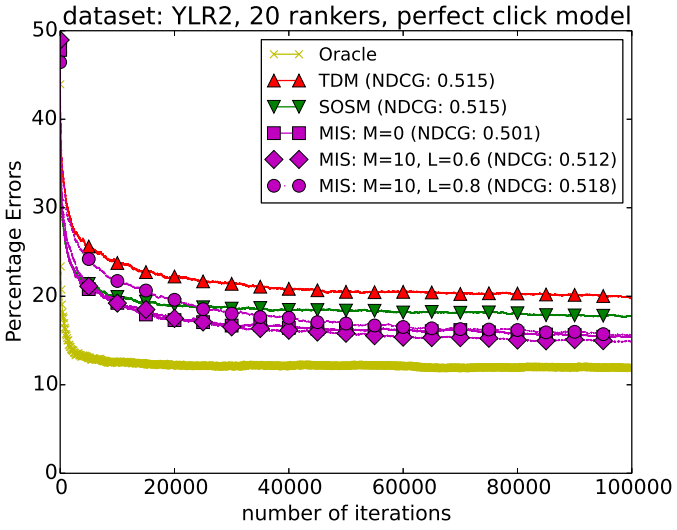


(C)

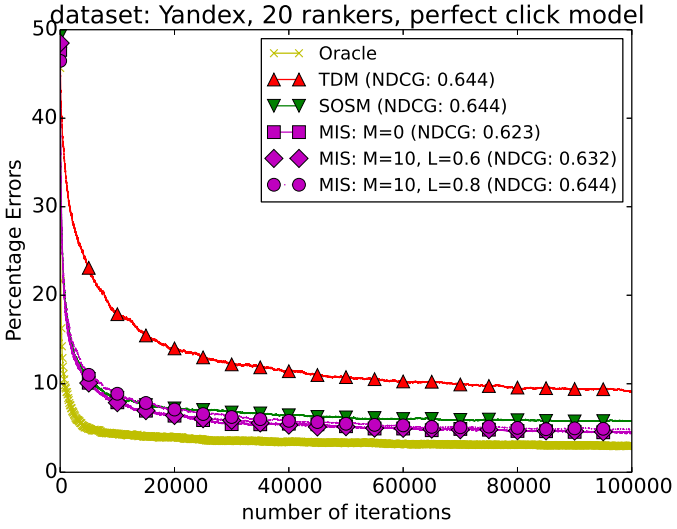
FIGURE 3.5: Percentage error averaged over 100 runs against number of iterations for different click models.



(A)

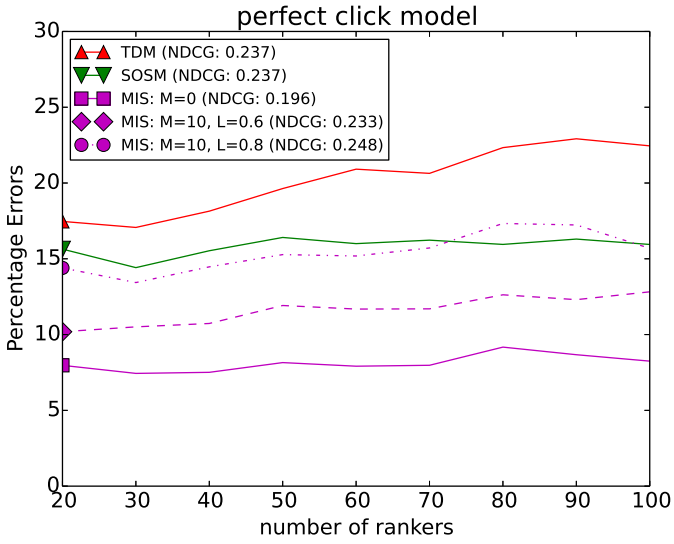


(B)



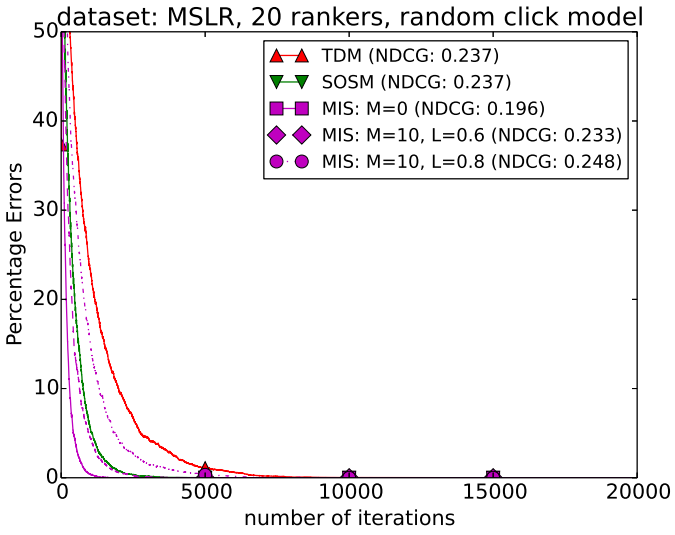
(C)

FIGURE 3.6: Percentage error averaged over 100 runs against number of iterations for different datasets. The figures show error rates for the YLR1, YLR2 and Yandex datasets with perfect click models.

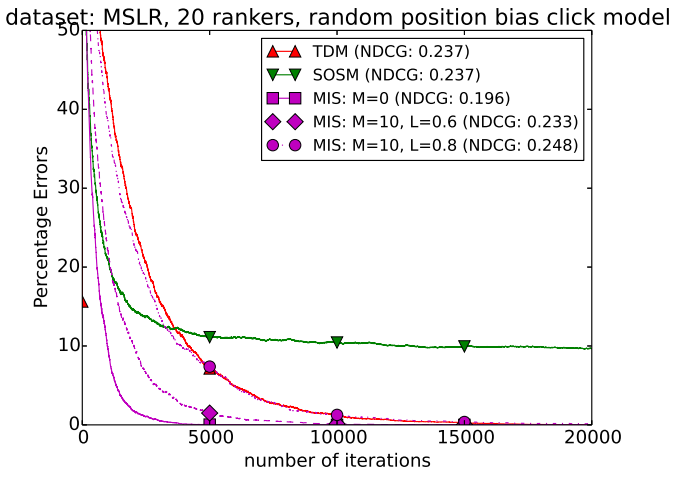


(A)

FIGURE 3.7: Percentage errors, averaged over 20 runs, after 20,000 iterations versus the number of rankers being compared on random subsets of k rankers for the MSLR dataset with perfect click model.



(A)



(B)

FIGURE 3.8: Percentage error against number of iterations for different click models. The figures show error rates for random click models with or without position bias

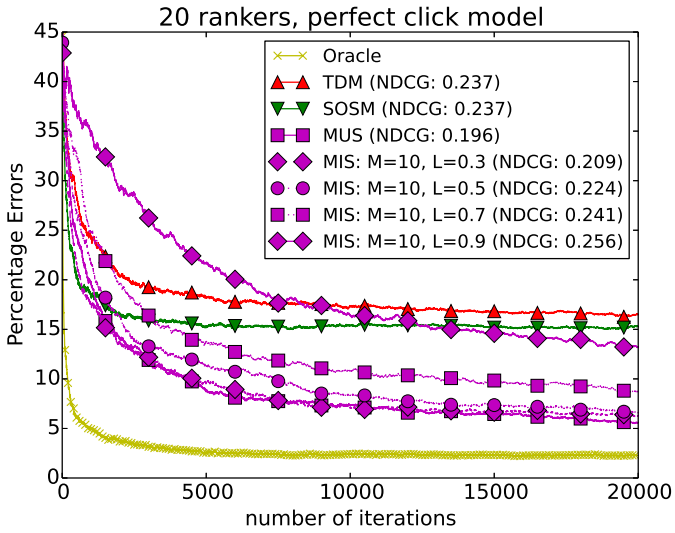


FIGURE 3.9: Percentage error against number of iterations for different parameter settings of MIS for the MSLR dataset with perfect click model.

Chapter 4

Online Learning

We now turn our attention to the second main problem of this thesis: How to manage the exploration-exploitation tradeoff associated with online evaluation using multileaving.

Recall that evaluation of rankers can be done online by presenting the ranked lists produced by rankers to users and then inferring the quality of the rankers by analyzing users' clicks and other forms of behaviour. Online evaluation of rankers has become increasingly popular, partly because user behaviour can be easily logged with no additional effort from the user. This provides online evaluation methods with inexpensive access to large amounts of timely training data [26]. One of the key drawbacks of online evaluation methods is that the outputs of new, potentially poor, rankers need to be presented to actual users. If a new ranker turns out to be poor, then users will be presented with poor results and, in the worst case, might abandon the service [54]. Conversely, if new rankers are not presented there is a risk of overlooking better rankers in the pool of rankers. In online learning the question of determining a proper exploration level is known as the *exploration-exploitation tradeoff*.

In online evaluation, it is usually easier for users to make relative judgements, rather than absolute judgements. For example, it is easier to say that document A is more relevant for a certain query than document B, than to say how relevant it is. Similarly, rankers A and B can be compared by *interleaving* their result lists and examining which documents a user clicks on. Interleaving methods were found to require 1-2 orders of magnitude less data than absolute metrics to detect even small differences in retrieval quality [26].

When using interleaving to compare pairs of rankers, it is critical to determine which two rankers to interleave at each comparison, i.e. to resolve the exploration-exploitation tradeoff. *Dueling bandits*, described in more detail in Section 2.4.2 is an elegant mathematical framework that provides a principled way for dealing with the exploration-exploitation trade-off in learning with relative preference feedback from pairwise comparisons [123].

More recently, interleaving has been generalized to *multileaving* which permits more than two rankers to be compared in a single comparison [102, 98, 17]. However this work focused only on the comparisons themselves, and did not address the key issue of selecting subsets of rankers for each comparison. This means that all rankers, both good and bad ones, were used in all the comparisons. This simple approach has several disadvantages. Firstly, since poor rankers are participating in all the comparisons the quality of the multileaved lists throughout the evaluation process is poor. And secondly, very poor rankers that could potentially be eliminated early in the

process continue being evaluated, which does not allow the comparisons to focus on rankers whose quality is harder to distinguish.

We extend the dueling bandit framework and propose a *Multi-Dueling Bandit* algorithm that provides an intelligent selection of rankers for simultaneous comparisons and improves the trade-off between exploration and exploitation. In Section 4.1 we introduce the Multi-Dueling Bandits setting and our algorithm. Here the observed outcomes are assumed to be pairwise wins. This is applicable to multileaving algorithms such as TDM and SOSM.

Unfortunately, as noted in Sections 2.3.5, these prior multileaving methods do not properly account for the interaction of the document sampling and ranker scoring phases. As a result they are prone to being inaccurate. In Section 3.3 we introduce MIS, a new multileaving method based on importance sampling, for which accuracy can be guaranteed. This multileaving method does not output pairwise wins, but instead provides score estimates for each ranker included in the multileaving. The Multi-Dueling Bandit problem setting is therefore not applicable. We instead introduce the Bandits with Multiple Plays setting in Section 4.2. Here we assume that the outcomes are absolute scores. This is applicable to our multileaving algorithm MIS. It can be regarded as an extension of the standard multiarmed bandit setting to multiple plays,

4.1 Multi-Dueling Bandits

Section 4.1.1 formalises the problem of learning with relative comparisons between multiple options as a K-armed *multi-dueling bandit* problem. Section 4.1.2 describes our proposed algorithm for solving this problem. Finally, Section 4.1.3 presents our experimental evaluation.

4.1.1 Problem Setting

In multi-dueling bandits, at each iteration, t, an algorithm selects a subset, S_t, of K arms and observes outcomes of noisy pairwise comparisons (duels) between all pairs of arms in S_t. In the ranking scenario this corresponds to multileaving the ranked lists of the subset, S_t, of rankers and then inferring the relative quality of the lists (and the corresponding rankers) from user clicks. When the size of S_t is limited to 2 the problem reduces to standard dueling bandits.

Let $P = [p_{ij}]$ be a matrix of probabilities that arm i wins in a pairwise comparison with arm j (it satisfies $p_{ij} = 1 - p_{ji}$ and we define $p_{ii} = \frac{1}{2}$). Recall that in pairwise comparisons the best arm is not always well-defined (recall the example with A being better than B, B better than C, and C better than A). We follow the assumption in most dueling bandit literature and assume that there exists a Condorcet winner, which is a unique arm $*$ satisfying $p_{*j} > \frac{1}{2}$ for all $j \neq *$. That is, the Condorcet winner $*$ is pairwise better than any other arm j. The quality of all arms is then defined by their *regret*, $r(j) = p_{*j} - \frac{1}{2}$, which is a shifted probability of losing to the best arm (this definition also coincides with dueling bandits). Smaller regret corresponds to better quality and the regret of playing the best arm is zero. The quality of a set of arms S_t is defined by the average quality of the constituent arms

(the average regret)

$$r(S_t) = \frac{\sum_{j \in S_t} p_{*j}}{|S_t|} - \frac{1}{2}. \tag{4.1}$$

The goal of a multi-dueling bandit algorithm is to select subsets of arms $S_1, S_2, \ldots$, so that the cumulative regret $\sum_{t=1}^{T} r(S_t)$ is minimized. All arms have to be selected a small number of times in order to be explored, but the goal of the algorithm is to minimize the number of times when suboptimal arms are selected. On average, simultaneous exploration has lower regret than sequential comparison.

Simultaneous comparison of more than two arms may affect their pairwise winning probabilities. For example, in ranking, the effective length of a multileaved ranked list is typically limited by 10 items, since users rarely go beyond the first page of results. Therefore, the simultaneous comparison of more than 10 rankers means that some rankers may be compared based on a merged list that does not include their top suggestions. This may affect the estimates of their relative quality. This effect, which we refer to as *distortion* may also occur when less than 10 rankers are compared, since the limited length of the merged list does not allow perfect representation of every ranker. The exact level of distortion depends on the data, ranker, and method used for multileaving. The level of distortion of estimates of the pairwise winning probabilities made by SOSM, which was used in our experiments, is evaluated in Section 3.3.3.2.5. It is important to emphasize that in all our experimental comparisons, except one pathological case, the advantage of parallel exploration outweighed the disadvantage due to distortion in estimates.

4.1.2 Algorithm

The proposed multi-dueling bandit algorithm is based on the principle of "optimism in the face of uncertainty" used in many other bandit algorithms. It maintains optimistic estimates of pairwise winning probabilities p_{ij} and plays arms that, according to these optimistic estimates, have a chance of being the Condorcet winner. When there is a single candidate, the algorithm *exploits* this knowledge and plays only that candidate. When there are multiple candidates the algorithm *explores* by comparing them all. We increase parallel exploration by adding additional arms to such comparisons, as described below.

Our estimates of pairwise winning probabilities are based on empirical counts of wins/losses. In order for these estimates to be meaningful the algorithm has to assume that pairwise winning probabilities are *consistent* with the pairwise winning probability matrix P, irrespective of the composition of the set S_t (meaning that they are not distorted). More precisely, since correct identification of the Condorcet winner depends on correct estimation of the probabilities p_{*j}, it is important that they remain at a certain margin above $\frac{1}{2}$ irrespective of the composition of S_t. Incorrect estimation of p_{ij}-s for $i, j \neq *$ does not influence identification of the Condorcet winner and, therefore, their distortion does not disturb the operation of the algorithm.

We now describe our algorithm, which is provided in the Algorithm 2 box. We denote by $n_{ij}(t)$ the number of times up to round t that i and j were compared with each other. Let $w_{ij}(t)$ denote the number of times when arm i beat arm j. We break ties randomly, so that $n_{ij}(t) = w_{ij}(t) + w_{ji}(t)$.

We compute upper confidence bounds $u_{ij}(t)$ on the probabilities p_{ij}:

$$u_{ij}(t) = \frac{w_{ij}(t)}{n_{ij}(t)} + \sqrt{\frac{\alpha \ln t}{n_{ij}(t)}} \tag{4.2}$$

(u_{ij}-s are the optimistic estimates of p_{ij}-s and they are analogous to those used in [132] for pairwise comparisons). The first term in $u_{ij}(t)$ is an empirical estimate of p_{ij} and the second term bounds the fluctuations of this estimate with high probability, see [6, 132]. The α parameter in the second term controls the width of the upper confidence bound.

Additionally, we maintain a second wider upper bound $v_{ij}(t)$, which we use to increase parallel exploration. We define $v_{ij}(t)$ by

$$v_{ij}(t) = \frac{w_{ij}(t)}{n_{ij}(t)} + \sqrt{\frac{\beta\alpha \ln t}{n_{ij}(t)}}, \tag{4.3}$$

where the parameter $\beta \geq 1$ controls how much wider it is than the upper confidence bound of Equation 4.2. When there is more than one candidate for a Condorcet winner according to the "narrow" confidence bounds in Equation 4.2 an exploration round is triggered and arms that could be Condorcet winner candidates according to the "wide" confidence bounds are compared. This leads to some arms being explored preemptively and decreases the overall number of exploration rounds by increasing parallel exploration.

Given K arms, we define $U_i(t) = \min_{j \in K, j \neq i} \{u_{ij}(t)\}$, i.e. $U_i(t)$ is the smallest upper confidence bound of i. Let E denote the set of potential Condorcet winners, which contains all arms i for which $U_i(t) \geq 1/2$. Additionally, we define $V_i(t) = \min_{j \in K, j \neq i} \{v_{ij}(t)\}$ and F to be the set of potential Condorcet winners according to the wider upper bounds, that is, all arms for which $V_i(t) \geq 1/2$.

At each iteration of Algorithm 2, if there is only a single potential Condorcet winner in E, we choose this arm. If there are several potential Condorcet winners, we select all arms in the larger set F. In the unlikely event that there are no potential Condorcet winners, we select all arms. The selected arms are compared against each other using multileaving and pairwise wins between the rankers are inferred from the scores produced by the multileaving method.

4.1.3 Experimental Evaluation

We next present the experimental evaluation of our Multi-Dueling Bandits (MDB) algorithm. We begin by describing our experimental setup.

We compare our MDB algorithm to three state-of-the-art dueling bandit algorithms, namely RUCB and MergeRUCB, both implemented in the freely available software package Lerot [100], and RMED1 [72]. As per [131], we set the α parameter for RUCB to 0.51, and to 1.01 for MergeRUCB. For RMED1 we use the same parameter setting as [72]: $f(K) = 0.3K^{1.01}$. To select the parameters for MDB, we carried out a grid search on the grid $\{0.5, 1, 1.5\} \times \{1.25, 1.5, 2, 4\}$ on a separate dataset, specifically the validation set of the YLR1 dataset, and found the best parameters to be $\alpha = 0.5$ and

$W = [w_{ij}] := 0_{K \times K}$
Play all arms and update the corresponding entries in W
for $t = 2, \ldots, T$ **do**
    $U := [u_{ij}(t)] = \frac{w_{ij}(t)}{n_{ij}(t)} + \sqrt{\frac{\alpha \ln t}{n_{ij}(t)}}$, $u_{ii}(t) = 1/2$
    $V := [v_{ij}(t)] = \frac{w_{ij}(t)}{n_{ij}(t)} + \sqrt{\frac{\beta\alpha \ln t}{n_{ij}(t)}}$, $v_{ii}(t) = 1/2$
    E = $\{i$ s.t. $U_i(t) \geq 1/2\}$ (The set of potential champions according to U)
    F = $\{i$ s.t. $V_i(t) \geq 1/2\}$ (The set of potential champions according to V)
    if $|E| > 1$ **then**
        Choose all arms $f \in F$ for comparison and update the corresponding entries in W
    else if $|E| = 1$ **then**
        Choose the arm $e \in E$
    else
        Choose all arms for comparison and update the corresponding entries in W
end

Algorithm 2: Multi-Dueling Bandit (MDB) Algorithm.

$\beta = 1.5$. We used these as our parameter settings for MDB for all other experiments.

We first compare the algorithms on artificial datasets where each arm has a utility which defines its winning probability against other arms, similar to the experiments proposed in [1]. In each iteration, for the arms chosen by the dueling or multi-dueling bandit algorithm, we sample from normal distributions with mean given by the utility of the arms, and unit variance to obtain scores for each arm. The arm utilities used are listed in Table 4.1. They were chosen to provide problem instances where the quality of the best arm was progressively less distinct from that of the other arms, and where the impact of increasing the number of arms could be isolated.

We also compare the algorithms on four large-scale evaluation datasets

TABLE 4.1: Datasets used for artificial utility based experiments.

Dataset	Distributions of Utilities of arms
1good5poor	1 arm with utility 0.8, 5 arms with utility 0.2
1good50poor	1 arm with utility 0.8, 50 arms with utility 0.2
1good200poor	1 arm with utility 0.8, 200 arms with utility 0.2
2good4poor	1 arm with utility 0.8, 1 arm with utility 0.7, 4 arms with utility 0.2
11good40poor	1 arm with utility 0.8, 10 arms with utility 0.7, 40 arms with utility 0.2
41good160poor	1 arm with utility 0.8, 40 arms with utility 0.7, 160 arms with utility 0.2
3good3poor	1 arm with utility 0.8, 2 arm with utility 0.7, 3 arms with utility 0.2
21good30poor	1 arm with utility 0.8, 20 arms with utility 0.7, 30 arms with utility 0.2
81good120poor	1 arm with utility 0.8, 80 arms with utility 0.7, 120 arms with utility 0.2
arith6	1 arm with utility 0.8, 5 arms with utilities forming arithmetic sequence between 0.7 and 0.2
arith51	1 arm with utility 0.8, 50 arms with utilities forming arithmetic sequence between 0.7 and 0.2
arith201	1 arm with utility 0.8, 200 arms with utilities forming arithmetic sequence between 0.7 and 0.2
geom6	1 arm with utility 0.8, 5 arms with utilities forming geometric sequence between 0.7 and 0.2
geom51	1 arm with utility 0.8, 50 arms with utilities forming geometric sequence between 0.7 and 0.2
geom201	1 arm with utility 0.8, 200 arms with utilities forming geometric sequence between 0.7 and 0.2

TABLE 4.2: Datasets. Each dataset consists of a number of query-document pairs, together with a relevance judgement for the pair. Each document is represented by a feature vector.

Datasets	Queries	URLs	Features
MSLR-WEB30K [2]	31,531	3,771,125	136
YLR Set 1 [25]	19,944	473,134	700
YLR Set 2 [25]	1,266	34,815	700
Yandex [3]	9,124	97,290	245

summarised in Table 4.2[1]. Since there was no Condorcet winner for the Yandex dataset, we randomly sampled subsets of 200 rankers from the Yandex dataset, selecting the first subset with a Condorcet winner. This subset was used in the experiments involving the Yandex dataset, except those described in Section 4.1.3.1.5, where we investigate the behaviour of the algorithms in the absence of a Condorcet winner.

The datasets and the corresponding rankers form our dueling or multi-dueling bandit problem instances. Following [131], for each dataset we choose the rankers to be the features of the dataset. That is, for a given feature, we construct a ranker which ranks documents only according to the score of that feature. An example of a feature is the BM25 score of the body of the document, or a document's PageRank. As noted in [131] this is a somewhat artificial setup from a learning-to-rank perspective, since we are generally interested in comparing different retrieval algorithms using all the features of the dataset, rather than finding the best individual feature. The benefit of this approach is that it makes the experiments easy to replicate. Furthermore, from the point of view of *evaluating* dueling and multi-dueling bandit algorithms, the difficulty of a problem instance is affected by the relative performance of the rankers, not their absolute performance. Using the feature rankers is therefore useful for assessing the performance of dueling and multi-dueling bandit algorithms since many of the features perform similarly and are therefore difficult to distinguish using interleaved or multileaved comparisons.

All experiments, except those using the artificial datasets described in Table 4.1, are conducted using a simulated user model. For each iteration we randomly sample with replacement one query from the pool of queries of the dataset. The dueling or multi-dueling bandit algorithms choose rankers, whose results are then interleaved or multileaved respectively, and presented to a simulated user. For the dueling bandit algorithms, we compare pairs of rankers using probabilistic interleaving [55], which is the best performing interleaving method to the best of our knowledge. For MDB, we use SOSM, which is the best performing multileaving method to the best of our knowledge [17]. Both probabilistic interleaving and SOSM only present the top-10 documents to users. This limit was chosen since it is rare for users to look

[1] Only 519 features are non-zero for YLR Set 1 and only 596 features are non-zero for YLR Set 2. The remaining features are zero for all query-document pairs.

TABLE 4.3: User Models. For each user model, the user inspects the ranked list from top to bottom and either clicks on a document, or stops inspecting the list, with each action's probability defined by the user model, and the document relevance. We use the perfect, navigational and informational click models from [56].

	Click Probabilities					Stop Probabilities				
Relevance	0	1	2	3	4	0	1	2	3	4
Perfect	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
Navigational	0.05	0.1	0.2	0.4	0.8	0.0	0.2	0.4	0.6	0.8
Informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

past the first page of results when using search engines. Clicks are then generated from a probabilistic user model [56]. The interleaving or multileaving algorithm scores the chosen rankers using the clicks generated by the user model.

The click model used for these experiments was the navigational user model from [56], unless otherwise stated. This click model describes a user who inspects the retrieved list of documents from top to bottom, and is more likely to click on a document if it is more relevant, but may interrupt their session with a certain probability, rather than inspect the entire list of retrieved documents. This click model has been used as a standard click model for dueling bandit algorithm evaluation in [131]. The click models are described in Table 4.3.

4.1.3.1 Experimental Results

Below we summarize the experimental results for the various experiments. For all figures, the error bars show the standard deviation of cumulative regret across runs for each algorithm at the given time step.

4.1.3.1.1 Experiments on synthetic data

We begin by examining how the cumulative regret increases at each iteration for each of the four algorithms. We use synthetic data for two reasons. First, synthetic data does not require interleaving or multileaving. This is because, at each iteration, after selecting the rankers to be compared, the comparison is performed based on drawing random numbers from a normal distribution with mean given by each arm's utility and unit variance. Thus, the performance of each of the four bandit algorithms is independent of the interleaving/multileaving, and only due to the bandit algorithm. The second reason for using synthetic data is that it allows us to control the relative performance of the individual arms. Clearly, if the best arm is much better that the other arm, the problem is easier than the case when the best arm is only slightly better than other arms.

Figure 4.1 shows the average cumulative regret against the number of iterations for the 4 algorithms on each of the artificial datasets from Table 4.1. MDB performs better than all the benchmark algorithms for all the datasets. The first column shows the datasets with 6 arms, the second

column shows the datasets with 51 arms and the third column shows the datasets with 201 arms. We see that while the regret does not increase noticeably for MDB as we increase the number of arms, the regret of all the dueling bandit algorithms increases substantially. For all the datasets with 51 or more rankers, MDB incurs at least an order of magnitude less regret than the best dueling bandit algorithm, RMED1.

The results demonstrate that as we increase the number of arms being compared, the advantages of MDB become larger. This advantage is at its most extreme when there is only one good arm, and all other arms are weaker, as in row 1 of Figure 4.1. In this case dueling bandit algorithms have to waste exploration time comparing suboptimal arms which are hard to differentiate from each other, and it can take a long time before the single good arm is identified.

4.1.3.1.2 Experiments on simulated Learning-to-Rank Datasets

Figure 4.2 shows how the cumulative regret increases with each iteration using the real data sets, MSLR, YLR1, YLR2 and Yandex. It clearly shows that MDB performs best for all the datasets. It outperforms the best dueling bandit algorithm, RMED1, by a factor of approximately 3 for the dataset with the smallest number of features, the MSLR dataset, and approximately 1-2 orders of magnitude for all the other datasets. RMED1 outperforms RUCB and MergeRUCB, as expected from the results of [72].

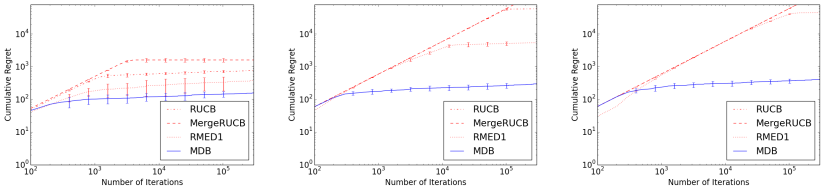
Note that for the Yandex dataset, since there was no Condorcet winner, we randomly sampled 200 of the 245 feature rankers to obtain a dataset with a Condorcet winner. Results using the full Yandex dataset with no Condorcet winner are described later.

4.1.3.1.3 Dependence on Number of Rankers

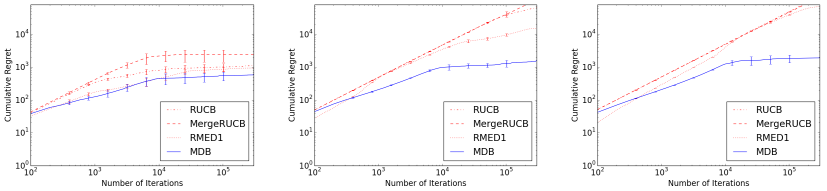
The results using synthetic data showed that the advantage of our algorithm increased relative to the three other algorithms, as the number of arms being compared increased. Additionally, for the results on the real learning-to-rank datasets, the advantage of our algorithm ranges from a factor of approximately 3 for the MSLR dataset with the smallest number of features to approximately 2 orders of magnitude for the two YLR datasets, which have the greatest numbers of features.

To isolate the impact of the number of rankers being compared on the real datasets involving multileaving, we investigate how regret scales with the number of rankers being compared using the YLR1 dataset. We randomly sampled subsets of rankers of sizes $\{10, 25, 40, 55, 70, 85, 100, 115, 130, 145\}$ from the YLR1 dataset. Note that we randomly sampled different subsets of rankers for each run. For each of these subsets we then carried out 10 runs of each algorithm over 5,000,000 iterations and recorded the average cumulative regret across runs.

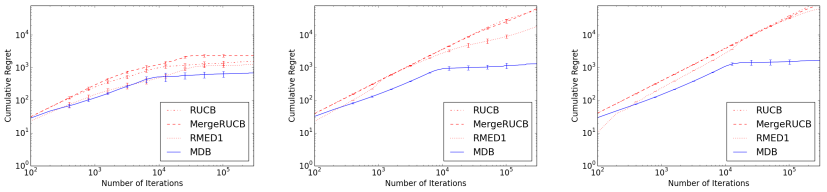
Figure 4.3 shows how the performance of the 4 algorithms varies as a function of the number of rankers. Additionally we have shown the performance of a random policy which simply selects a random subset of the rankers for multileaving at each iteration. We observe that as the number of rankers increases the cumulative regret increases most for RUCB and MergeRUCB, while it increases more slowly for RMED1. Regret appears to be almost independent of the number of rankers for MDB.



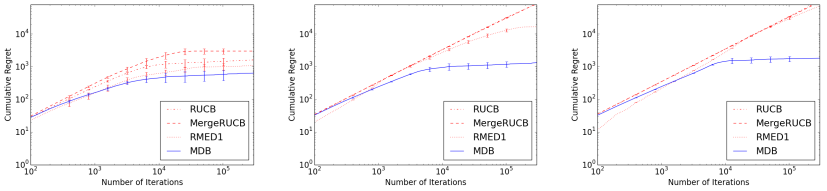
1good5poor 1good50poor 1good200poor



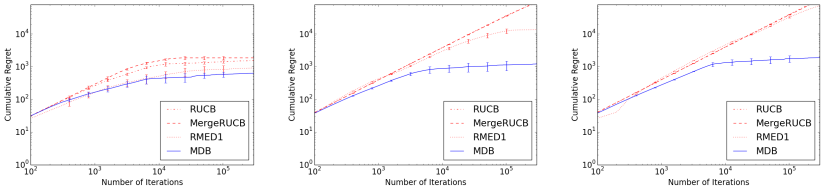
2good4poor 11good40poor 41good160poor



3good3poor 21good30poor 81good120poor

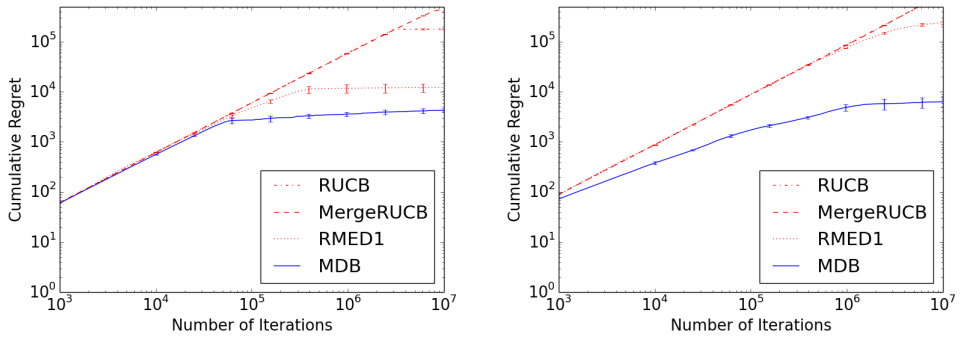


arith6 arith51 arith201



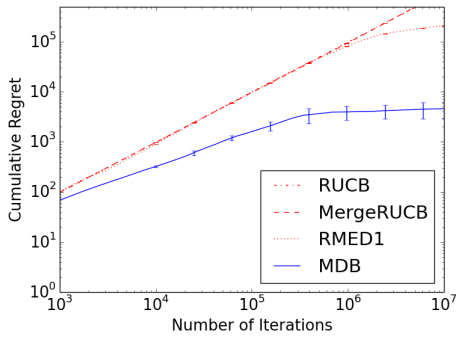
geom6 geom51 geom201

FIGURE 4.1: Cumulative regret averaged over 10 runs against number of iterations for the 4 algorithms on the datasets listed in Table 4.1

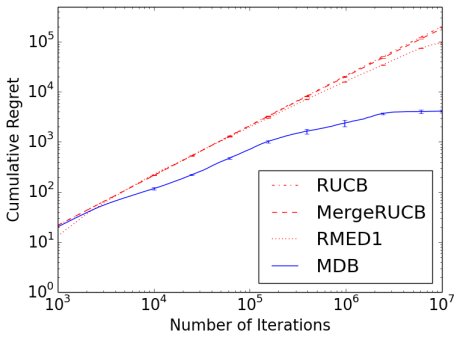


(A) MSLR

(B) YLR1



(C) YLR2



(D) Yandex

FIGURE 4.2: Cumulative regret averaged over 10 runs against number of iterations for the 4 algorithms on the MSLR (a), YLR1 (b), YLR2 (c) and Yandex (d) datasets with the navigational click model.

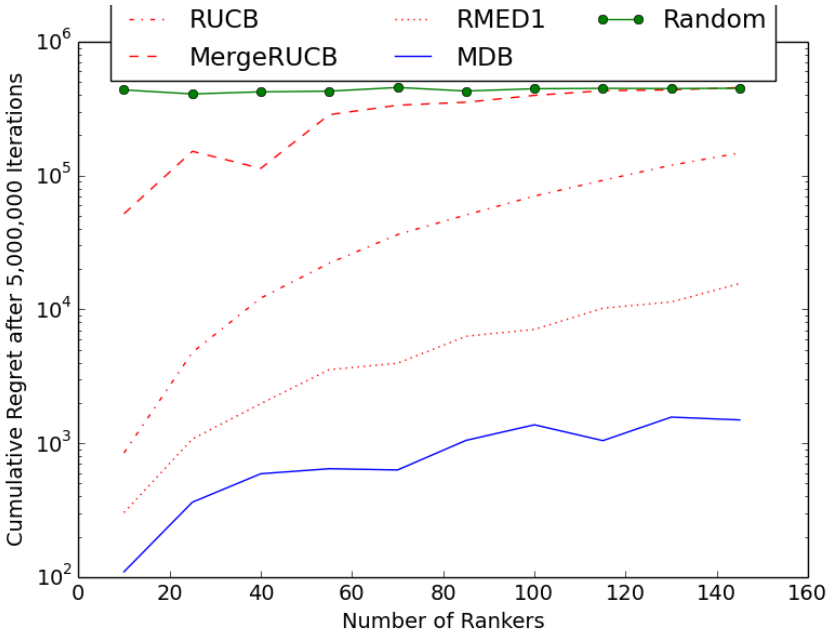


FIGURE 4.3: Cumulative regret averaged over 10 runs after 5,000,000 iterations against number of rankers for the 4 algorithms, and a random policy, on subsets with M rankers of the YLR1 dataset with navigational click model.

These experiments were also carried out for the perfect and informational click models, and the results were very similar.

For the MDB algorithm, the regret associated with having to explore suboptimal rankers does not appear to be additionally compounded by the number of rankers being explored. This is an important characteristic of the MDB algorithm, since if we can explore additional rankers with no substantial additional cost, the risks associated with large-scale online ranker evaluation are substantially mitigated.

Note that it may appear that regret levels off for MergeRUCB as we increase the number of rankers. This is due to the fact that for 5,000,000 iterations there is a limit to how much regret can be incurred just by making random choices in 5,000,000 iterations. For larger problem sizes and for a time frame of 5,000,000 iterations, MergeRUCB begins to perform no better than a random policy. This does not imply that MergeRUCB performs as badly as a random policy in general, but for these problem instances it has not yet begun to eliminate suboptimal arms after 5,000,000 iterations. Further iterations would be needed to show improvements relative to the random policy.

4.1.3.1.4 Dependence on Click Model

To test the robustness of our approach to the choice of click model, we also investigated performance using the perfect and informational click models [56]. These click models have less, respectively more, noisy user behaviour than the navigational model. Since different click models can reflect different types of user behaviour and search intent it is important that the algorithms are robust to different click models. Figure 4.4 shows how the cumulative

regret is affected by different user click models, using randomly selected subsets of size 200 of the rankers from the YLR1 dataset. We chose to use subsets of the full dataset for these experiments because of the computational costs of running RMED1 on the full YLR1 dataset. For all click models MDB outperforms the best dueling bandit algorithm by between 1 and 2 orders of magnitude.

For MDB, the regret doubles when going from the perfect to the navigational click model, but does not increase further for the informational click model. In contrast, for the dueling bandit algorithms, regret for the informational click model is approximately double that for the navigational click model, which is approximately double that of the perfect click model. MDB is therefore least affected by varying the click model in our experiments.

4.1.3.1.5 Dataset without Condorcet winner

As noted earlier, the baseline dueling bandit algorithms and our algorithm MDB assume the existence of a Condorcet winner, i.e. a ranker which beats every other ranker in expectation. In practice, this may not be true, and in fact there is no Condorcet winner for the full Yandex dataset. To evaluate how the algorithms perform when the Condorcet assumption is violated, we investigated the performance of the algorithms on the full Yandex dataset. Since there is no Condorcet winner we cannot use the regret definition from Equation 4.1 to evaluate the algorithms. Instead we define the winner for the full Yandex dataset based on the NDCG@10 score, denote this score by $NDCG^*$, and use a definition of regret given by

$$r(S_t) = \frac{\sum_{j \in S_t} NDCG^* - NDCG_j}{|S_t|}. \tag{4.4}$$

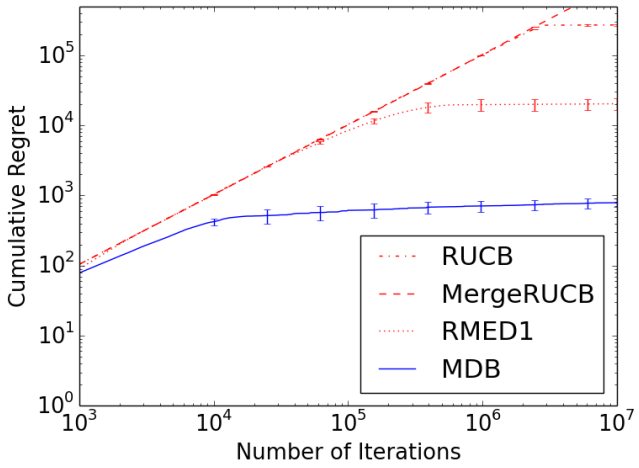
We carried out 10 runs of each algorithm over 5,000,000 iterations and recorded the average regret over runs at each iteration. Figure 4.5 shows how the cumulative regret increases with each iteration for the full Yandex dataset with regret defined in Equation 4.4. The results are very similar to those from Figure 4.2 for the Yandex subset with a Condorcet winner. MDB outperforms the best dueling bandit algorithm, RMED1, by approximately an order of magnitude after 5,000,000 iterations.

4.1.3.1.6 Distortion of probability estimates due to multileaving

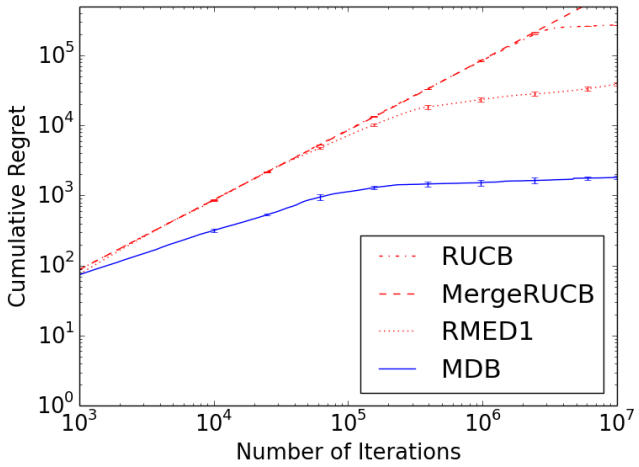
As discussed earlier, simultaneous comparison of more than two arms may affect their pairwise winning probabilities. We called this effect *distortion*. We can quantify this effect by first randomly sampling a fixed size subset of rankers that includes a Condorcet winner, and then measuring, after some fixed number of multileavings, the fraction of rankers that beat the Condorcet winner more than 50% of the time. If there is no distortion, and the number of multileavings is sufficient, we expect this fraction to be zero.

In these experiments we test the level of distortion in the multileaving method SOSM, and examine how robust our MDB algorithm is to possible distortions in the multileaving method.

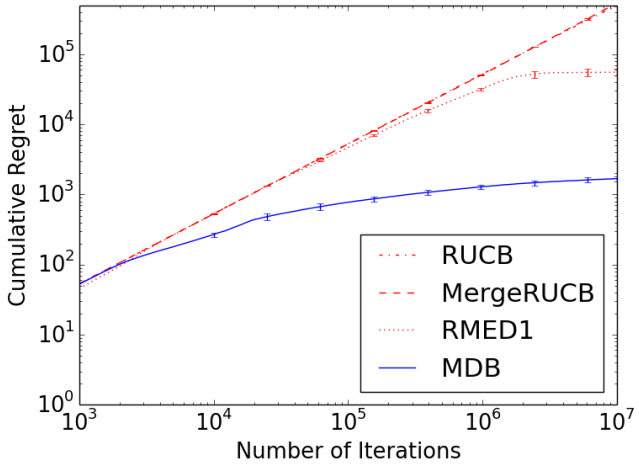
For each dataset, and each click model, we randomly sample subsets of rankers of sizes 3, 10, and 100 that include a Condorcet winner. We examine the probabilities of the rankers beating the Condorcet winner after 3,000



(A) perfect click model



(B) navigational click model



(C) informational click model

FIGURE 4.4: Cumulative regret averaged over 10 runs against number of iterations for the 4 algorithms on the YLR1 dataset using the (a) perfect, (b) navigational and (c) informational click models.

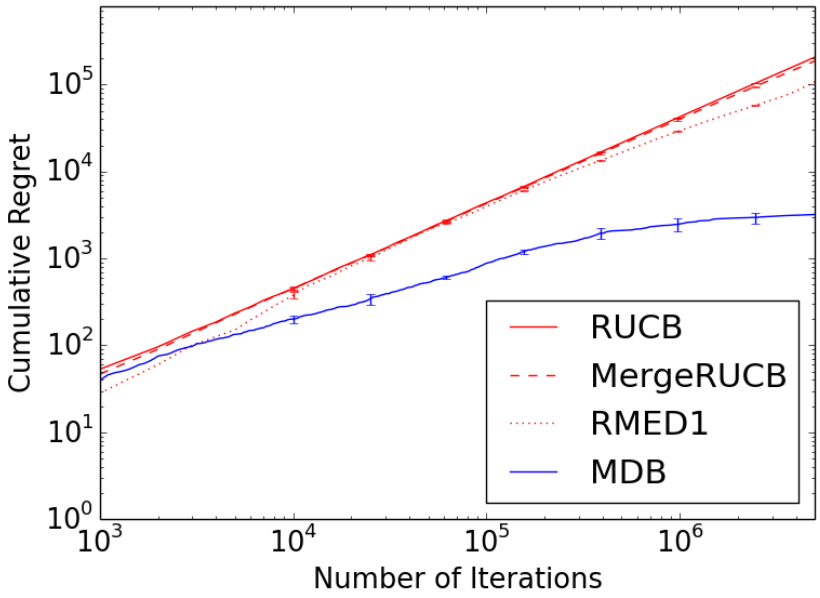


FIGURE 4.5: Average cumulative regret over 10 runs against number of iterations for the 4 algorithms on the full Yandex dataset.

multileavings. Note that this is likely to be an overestimate of the distortion of the multileaving method, since, for rankers of very similar quality, 3,000 iterations may not be sufficient to reliably distinguish rankers. Table 4.4 shows the average, over 30 runs, of the percentage of rankers that beat the Condorcet winner.

We observe that the distortion problem is almost unique to the MSLR dataset, and is exacerbated by the noisier click models. The distortion problem is exclusively related to the feature ranker 133 in the MSLR dataset. Feature ranker 133 scores documents solely based on the query-document clicks, i.e. a document was clicked on in response to a query. This feature is very good at identifying 1 or 2 documents that are very likely to be relevant. However, when asked to rank documents in a multileaved set, most of the documents, even though they might be relevant, have not been previously clicked on. As such, ranker 133 is unable to distinguish between the vast majority of documents. Thus, even though ranker 133 performs well in pairwise comparisons, where it has contributed half of the documents in the results list, it performs very poorly when multileaved with many other rankers. Table 4.4 also includes results for the MSLR dataset, when feature ranker 133 is excluded. This is denoted by MSLR*. When ranker 133 is excluded, no substantial distortion is observed.

The moderate levels of distortion observed for the Yandex and MSLR dataset (excluding feature ranker 133) are likely to be mostly caused by the fact that there are many rankers that are very similar in quality, and so 3,000 comparisons are not sufficient to differentiate these similar rankers.

The only problem setting where our MDB algorithm did not substantially outperform the best baseline dueling bandit algorithm, RMED1, was for the MSLR dataset with all 136 feature rankers with the informational click model. The results for this problem setting are shown in Figure 4.6.

TABLE 4.4: Percentage of rankers beating the Condorcet winner (distortion), averaged over 30 runs, after 3,000 iterations for 3, 10, and 100 rankers being multileaved for each dataset and click model. The dataset denoted MSLR* is the MSLR dataset with feature ranker 133 removed.

	Distortion		
Num. Rankers	3	10	100
MSLR Perfect	0.0%	0.0%	9.2%
MSLR Navigational	0.0%	0.0%	15.2%
MSLR Informational	0.0%	6.8%	41.3%
MSLR* Perfect	1.7%	3.1%	3.3%
MSLR* Navigational	1.7%	4.0%	3.5%
MSLR* Informational	0.0%	2.9%	2.7%
YLR1 Perfect	0.0%	0.0%	0.0%
YLR1 Navigational	0.0%	0.0%	0.0%
YLR1 Informational	0.0%	0.0%	0.3%
YLR2 Perfect	0.0%	0.0%	0.4%
YLR2 Navigational	0.0%	0.4%	0.8%
YLR2 Informational	0.0%	1.0%	0.9%
Yandex Perfect	0.0%	1.1%	1.4%
Yandex Navigational	3.3%	4.5%	3.8%
Yandex Informational	3.3%	3.9%	3.4%

This is due specifically to the feature ranker 133 in the MSLR dataset. Table 4.4 shows that there was some distortion for all click models for the MSLR dataset. However, it is with the informational click model that the distortion is greatest, reaching 41.3% for 100 rankers. This is a very high percentage. The MDB algorithm appears to be robust to more reasonable levels of distortion, suffering substantially less regret than the baselines for the MSLR dataset with the navigational click model, shown in Figure 4.2(a), and with the perfect click model. Additionally, for the MSLR dataset with feature ranker 133 removed, MDB substantially outperformed all baselines for all click models.

4.1.4 Conclusions

We proposed a generalisation of the K-armed dueling bandits, termed multi-dueling bandits (MDB). We have applied MDB in online ranker evaluation to leverage the power of simultaneous comparisons through multileaving and

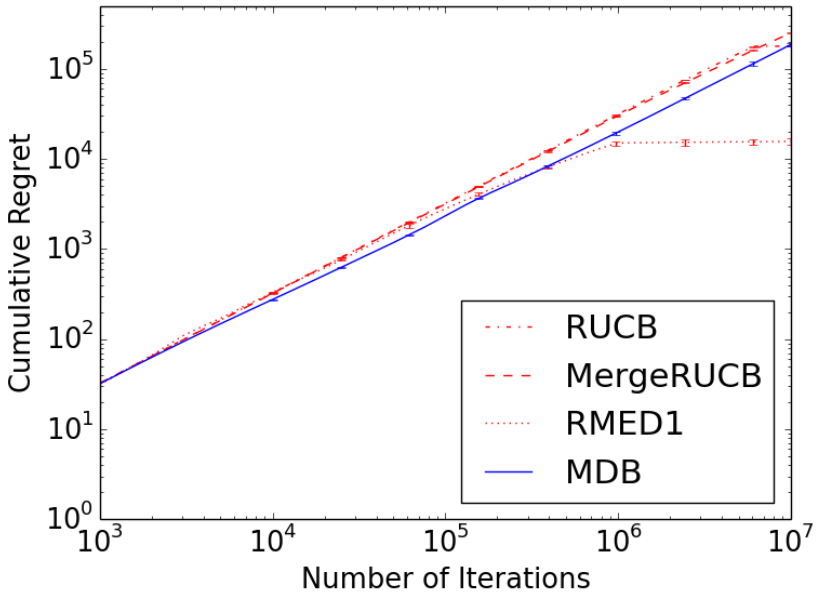


FIGURE 4.6: Average cumulative regret over 10 runs against number of iterations for the 4 algorithms on the MSLR dataset using the informational click model.

improve the exploration-exploitation trade-off. Our experimental results on synthetic data and data from 4 standard datasets demonstrated up to 1 to 2 orders of magnitude reduction in regret compared to state-of-the-art dueling bandit algorithms, RUCB [132], MergeRUCB [131], and RMED1 [72] in all except one pathological case discussed below. Generally, relative benefits compared to dueling bandits increased with the number of rankers being compared. For MDB, the incurred regret did not increase substantially as the number of rankers increased. As such, the risks associated with large-scale online ranker evaluation are substantially mitigated. Further experiments showed that MDB was robust to various user click models.

Experiments were also conducted to examine the behaviour of MDB in the absence of a Condorcet winner, which is the case for the full Yandex dataset. In this case, the regret was approximated by measuring the NDGC@10 score. In this case MDB outperformed the best dueling bandit algorithm, RMED1, by approximately an order of magnitude after 5,000,000 iterations.

We also investigated the level of distortion of pairwise winning probabilities in multileaving using SOSM. For the MSLR dataset using a navigational click model, the distortion reached 41.3%. In this case MDB was inferior to RMED1. The high level of distortion was due to the peculiarities of ranker 133. If ranker 133 is removed, the distortion of pairwise winning probabilities is significantly reduced and MDB outperforms all other algorithms.

There are a number of avenues for future work. The distortion of pairwise winning probabilities in multileaving needs further investigation. All existing multileaving algorithms exhibit this behaviour. It remains an open question as to whether a new multileaving algorithm can be designed to avoid this problem, or at least minimize it. Furthermore, a theoretical analysis of our algorithm needs to be developed to better understand its power and

limitations. Additionally, since a Condorcet winner is not guaranteed to exist, it may be useful to explore other concepts of winners, such as the Copeland [130], Borda [117] and von Neumann [42] criteria. Finally, we note that the proposed multi-dueling bandit algorithm can be applied to a broad class of problems and applications in other domains, e.g. recommender systems.

4.2 Bandits with Multiple Plays

Section 4.2.1 formalises the problem of learning from multileaved comparisons which produce absolute scores as a K-armed *bandits with multiple plays* problem. Section 4.2.2 describes our proposed algorithm for solving this problem. Finally, Section 4.2.3 presents our experimental evaluation.

4.2.1 Problem Setting

In bandits with multiple plays, we are given a set $\mathcal{K}$ of arms. At each iteration, t, an algorithm selects a subset, S_t, of arms and observes rewards in the range $[0, b]$ for each selected arm $i \in S_t$. These rewards are independently and identically distributed according to corresponding distributions with unknown expectations μ_i.

In the ranking scenario this corresponds to multileaving the ranked lists of the subset, S_t, of rankers and then inferring a score for each ranker from user clicks.

We let $\mu^* = \max_i \mu_i$, and let $\Delta_i = \mu^* - \mu_i$. Then we define the regret

$$r(S_t) = \frac{\sum_{i \in S_t} \Delta_i}{|S_t|} \tag{4.5}$$

The goal of a bandits with multiple plays algorithm is to select subsets of arms $S_1, S_2, \ldots$, so that the cumulative regret $\sum_{t=1}^{T} r(S_t)$ is minimized.

4.2.2 Algorithm

The proposed bandits with multiple plays algorithm has a similar underlying idea to our multi-dueling bandit algorithm from Section 4.1.

The algorithm maintains optimistic and pessimistic estimates of the expected value of each arm. Furthermore, it maintains wide versions of these estimates. We therefore maintain *upper confidence bounds*, *wide upper confidence bounds*, *lower confidence bounds* and *wide lower confidence bounds* for the expected value of each arm. The wide confidence bounds serve to ensure that we explore arms in parallel. How this is done will be explained below.

At a given iteration, we say that an arm is a *champion* if it has the greatest upper confidence bound. If there are several arms with the same upper confidence bound we choose one of them at random to be champion. We call those arms whose upper confidence bound exceed the lower confidence bound of the champion *candidates*. Additionally, we call those arms whose wide upper confidence bounds exceed the wide lower confidence bound of the champion *secondary candidates* When there is a single champion, and no other candidates, the algorithm exploits this knowledge and plays only

the champion. When there are multiple candidates the algorithm explores by selecting all the secondary candidates.

We now precisely describe our algorithm, which is provided in the Algorithm 3 box. Let K be the number of arms. We denote by $n_i(t)$ the number of times up to round t that arm i was played. Let $\bar{x}_i(t)$ denote the mean score of arm i after t iterations, and $\bar{y}_i(t)$ denote the variance of the scores of arm i after t iterations. Recall that the scores for all arms have support $[0, b]$. We compute upper and lower confidence bounds $u_i(t)$ and $l_i(t)$ on the scores $\bar{x}_i(t)$:

$$u_i(t) = \bar{x}_i(t) + \sqrt{\frac{2\alpha\bar{y}_i(t)\ln t}{n_i(t)}} + \frac{3b\alpha\ln t}{n_i(t)} \tag{4.6}$$

$$l_i(t) = \bar{x}_i(t) - \sqrt{\frac{2\alpha\bar{y}_i(t)\ln t}{n_i(t)}} - \frac{3b\alpha\ln t}{n_i(t)} \tag{4.7}$$

Note that we use variance estimates in our confidence bounds, this should be advantageous if the variances of the scores of the arms are substantially smaller than b^2 [5]. These confidence bounds are based on an empirical Bernstein bound, see [4].

They ensure that with high probability the real expected value μ_i of the score of arm i is in the range $[l_i(t), u_i(t)]$.

Additionally, we maintain a second wider set of bounds $v_i(t)$ and $m_i(t)$, which we use to increase parallel exploration. We define them by

$$v_i(t) = \bar{x}_i(t) + \sqrt{\frac{2\beta\alpha\bar{y}_i(t)\ln t}{n_i(t)}} + \frac{3b\beta\alpha\ln t}{n_i(t)} \tag{4.8}$$

$$m_i(t) = \bar{x}_i(t) - \sqrt{\frac{2\beta\alpha\bar{y}_i(t)\ln t}{n_i(t)}} - \frac{3b\beta\alpha\ln t}{n_i(t)} \tag{4.9}$$

where the parameter $\beta \geq 1$ controls how much wider they are than the upper confidence bounds of Equation 4.7. When there is more than one candidate according to the "narrow" confidence bounds in Equation 4.7 an exploration round is triggered and the secondary candidates according to the "wide" confidence bounds are selected. This leads to some arms being explored preemptively and decreases the overall number of exploration rounds by increasing parallel exploration.

4.2.3 Experimental Evaluation

We next present the experimental evaluation of our bandits with multiple plays (BMP) algorithm. We begin by describing our experimental setup.

4.2.3.1 Experimental Setup

Since this is a new problem setting, there are no prior algorithms against which we can directly compare our algorithm. However since the intended application for our algorithm is online evaluation of rankers using multileaving, we will compare against the MDB algorithm [18] from Section 4.1. We set the parameters for MDB to $\alpha = 0.5$ and $\beta = 1.5$, as in Section 4.1.

$X = [\bar{x}_i(t)] := 0_K$
Play all arms and update the corresponding entries in X
for $t = 2, \ldots, T$ **do**
  $U := [u_i(t)] = \bar{x}_i(t) + \sqrt{\frac{2\alpha\bar{y}_i(t)\ln t}{n_i(t)}} + \frac{3b\alpha\ln t}{n_i(t)}$
  $V := [v_i(t)] = \bar{x}_i(t) + \sqrt{\frac{2\beta\alpha\bar{y}_i(t)\ln t}{n_i(t)}} + \frac{3b\beta\alpha\ln t}{n_i(t)}$
  $L := [l_i(t)] = \bar{x}_i(t) - \sqrt{\frac{2\alpha\bar{y}_i(t)\ln t}{n_i(t)}} - \frac{3b\alpha\ln t}{n_i(t)}$
  $M := [m_i(t)] = \bar{x}_i(t) - \sqrt{\frac{2\beta\alpha\bar{y}_i(t)\ln t}{n_i(t)}} - \frac{3b\beta\alpha\ln t}{n_i(t)}$
  c = argmax$_{i\in\mathcal{K}}(U_i(t))$ (The champion. If there are several arms, choose one at random)
  D = $\{i$ s.t. $U_i(t) \geq L_c(t)\}$ (The set of candidates)
  E = $\{i$ s.t. $V_i(t) \geq M_c(t)\}$ (The set of secondary candidates)
  if $|D| > 1$ **then**
    Choose all arms in E
  else
    Choose c
end

Algorithm 3: Bandits with Multiple Plays (BMP).

For MDB we use SOSM, described in Section 3.2 as the input multileaving method.

To select the parameters for BMP, we carried out a grid search on the grid $\{0.01, 0.05, 0.5, 1.5\} \times \{1.25, 1.5, 2.0, 4.0, 8.0, 16.0\}$ on a separate dataset, specifically the validation set of the YLR1 dataset, and found the best parameters to be $\alpha = 0.05$ and $\beta = 4.0$. For BMP we use MIS, described in Section 3.3 as the input multileaving method. For MIS we used the parameter settings: $M = 10, L = 0.6$, which were also found to be the best out of the options $M = 0$; $M = 10, L = 0.6$; and $M = 10, L = 0.8$ on the validation set of the YLR1 dataset.

Since the multileaving methods used as inputs to MDB and BMP are different, the quality of the multileaved lists presented to users is different, even when the same set of rankers is multileaved. For the experiments to follow we therefore use the following notion of regret so that we are comparing methods in terms of the quality of the multileaved lists they display to users:

$$r(S_t) = NDCG^* - NDCG^{M_t}, \tag{4.10}$$

where $NDCG^*$ is the average NDCG@10 score of the ranker with the best NDCG@10 score on the dataset, and $NDCG^{M_t}$ is the NDCG@10 score of the multileaved list presented to the user at iteration t.

Otherwise, we replicate the experimental setup from Section 4.1.3. We compare the algorithms on the four large-scale evaluation datasets summarised in Table 4.5[4].

The datasets and the corresponding rankers form our BMP or MDB problem instances. For each dataset we choose the rankers to be the features of the dataset. All experiments are conducted using a simulated user model.

[4]Only 519 features are non-zero for YLR Set 1 and only 596 features are non-zero for YLR Set 2. The remaining features are zero for all query-document pairs.

TABLE 4.5: Datasets. Each dataset consists of a number of query-document pairs, together with a relevance judgement for the pair. Each document is represented by a feature vector.

Datasets	Queries	URLs	Features
MSLR-WEB30K [5]	31,531	3,771,125	136
YLR Set 1 [25]	19,944	473,134	700
YLR Set 2 [25]	1,266	34,815	700
Yandex [6]	9,124	97,290	245

TABLE 4.6: User Models. For each user model, the user inspects the ranked list from top to bottom and either clicks on a document, or stops inspecting the list, with each action's probability defined by the user model, and the document relevance. We use the perfect, navigational and informational click models from [56].

	Click Probabilities					Stop Probabilities				
Relevance	0	1	2	3	4	0	1	2	3	4
Perfect	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
Navigational	0.05	0.1	0.2	0.4	0.8	0.0	0.2	0.4	0.6	0.8
Informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

For each iteration we randomly sample with replacement one query from the pool of queries of the dataset. The BMP or MDB algorithms choose rankers, whose results are then multileaved and presented to a simulated user. Clicks are then generated from a probabilistic user model [56] described in Table 4.6.

4.2.3.2 Experimental Results

Below we summarize the experimental results for the various experiments. The error bars show the standard deviation of cumulative regret across runs for each algorithm at the given time step.

4.2.3.2.1 Click Models

Figure 4.7 shows the performance of the BMP and MDB algorithms on the MSLR dataset for the perfect, navigational and informational click models. We note that for all click models, BMP outperforms MDB. While the difference is relatively small for the perfect and navigational click models, for the informational click model, MDB settles on the wrong arm resulting in linear regret. As a result, the regret suffered by BMP is almost 2 orders of magnitude less than that suffered by MDB after 10,000,000 iterations. This tendency of MDB to settle on the wrong arm can be understood in light of

the distortion suffered by the multileaving method used by MDB, SOSM, described further in Section 4.1.3.1.6.

4.2.3.2.2 Datasets

Figure 4.8 shows the performance of the BMP and MDB algorithms on the MSLR dataset for the YLR1, YLR2 and Yandex datasets with navigational click model. For all three datasets the algorithms perform similarly in terms of cumulative regret after 10,000,000 iterations.

Note, however, that for the YLR2 dataset, although BMP has suffered more regret after 10,000,000 iterations, the regret does not really level off for MDB. This is because the Condorcet winner, which MDB identifies as the best ranker and converges on selecting, is actually marginally inferior in terms of NDCG to another ranker. This ranker is correctly identified by BMP, even though BMP takes longer to converge on selecting it. If the time horizon for the experiment had been sufficiently long, BMP would therefore have suffered lower regret than MDB for this setting.

4.2.3.2.3 Scalability

Finally, Figure 4.9 shows how the performance of BMP and MDB scales with the number of rankers being evaluated. We note that the regret suffered after 5,000,000 iterations increases only weakly as we increase the number of rankers for both algorithms. This suggests that the parallel exploration enabled by the BMP and MDB settings is effective at reducing the dependency of the regret on the number of rankers being evaluated.

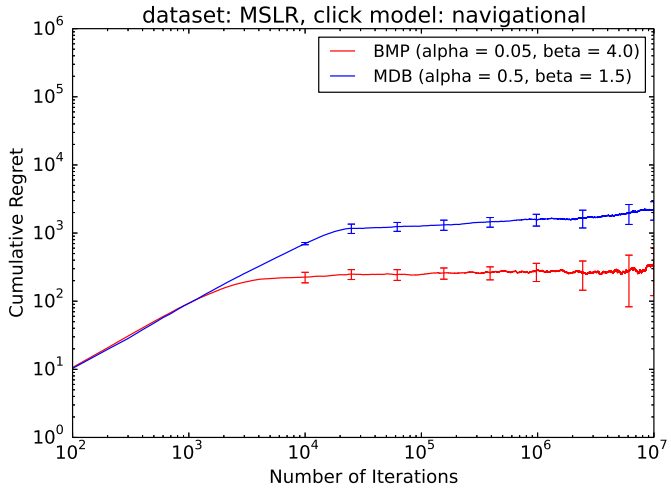
4.2.4 Conclusions

We have proposed a variant of the bandit with multiple plays (BMP) problem where the number of arms selected at each iteration is not fixed. This is applicable to managing the exploration-exploitation tradeoff associated with a multileaving algorithm like MIS, introduced in Section 3.3, where the outcome of a multileaving is an absolute score for each ranker in the comparison set.

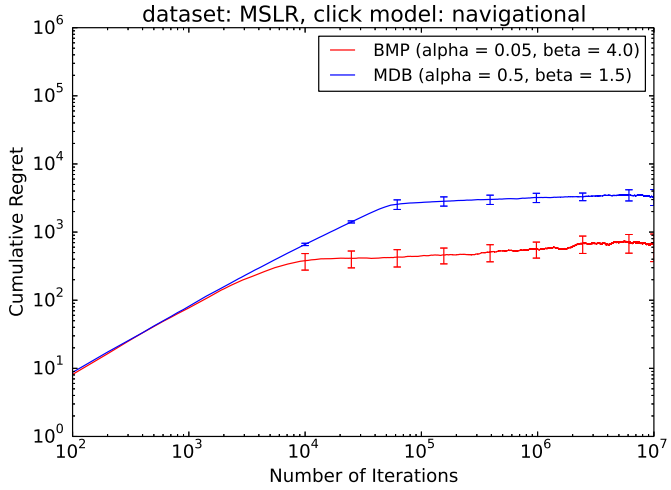
Our experimental results on data from 4 standard datasets demonstrated improved, or similar performance, relative to MDB for all datasets and click models investigated. Most notably, while BMP is never substantially outperformed by MDB, it substantially outperforms MDB for the MSLR dataset. Like MDB, the regret suffered by BMP scales well with the number of rankers being evaluated.

Thus, BMP preserves the strength of MDB, in providing a highly scalable method for handling the exploration-exploitation tradeoff associated with online evaluation of rankers using multileaving. Additionally, BMP provides the advantage of being applicable to a more accurate multileaving algorithm than those which MDB is applicable to.

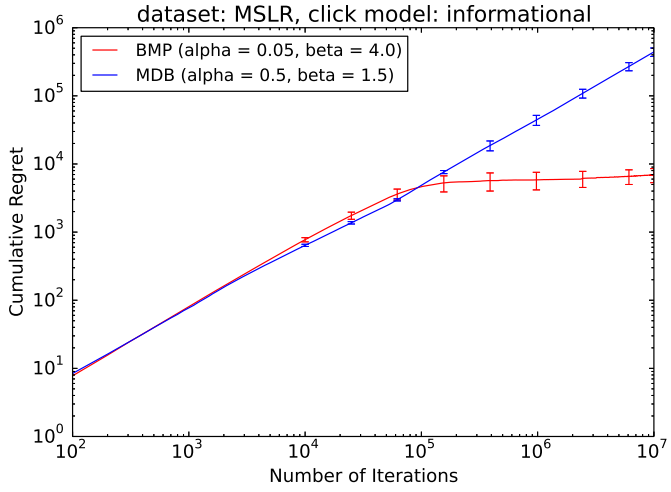
An important future extension of this work is to develop upper and lower bounds on the regret attainable in the BMP setting. Additionally, it is likely that improved regret can be obtained by leveraging dependencies between the rewards for different arms as in [107].



(A)

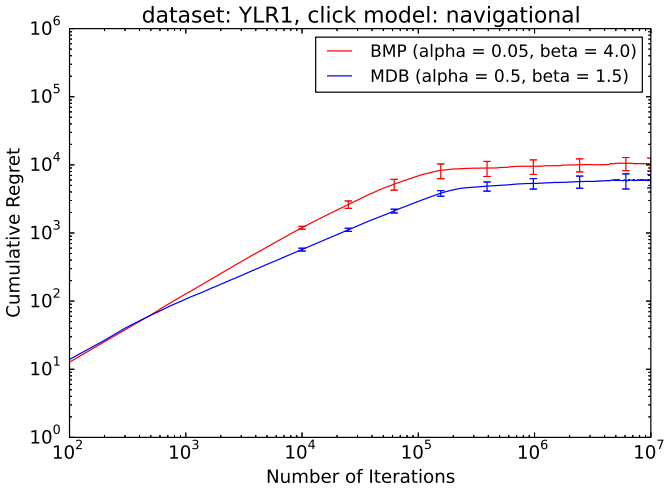


(B)

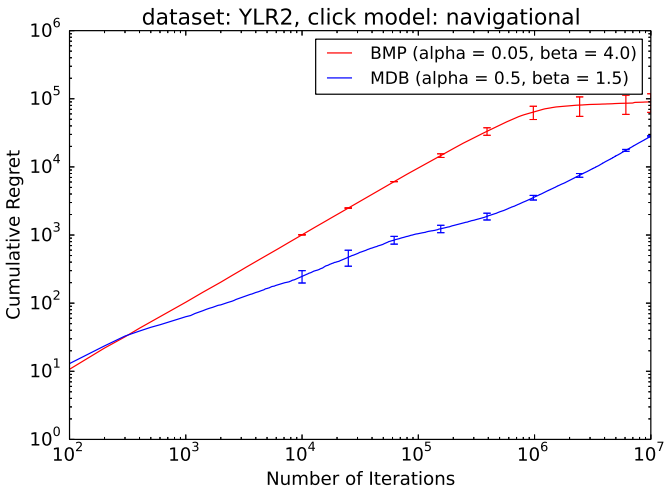


(C)

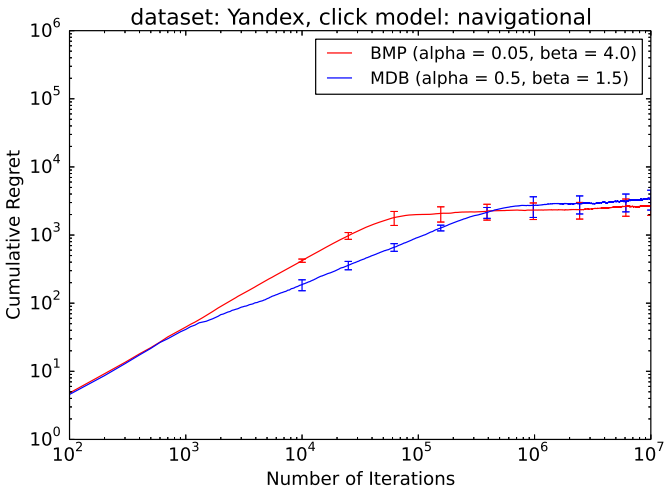
FIGURE 4.7: Cumulative regret averaged over 20 runs against number of iterations for the (a) perfect, (b) navigational, and (c) informational click models on the MSLR dataset.



(A)



(B)



(C)

FIGURE 4.8: Cumulative regret averaged over 20 runs against number of iterations for the (a) YLR1, (b) YLR2, and (c) Yandex datasets with navigational click model.

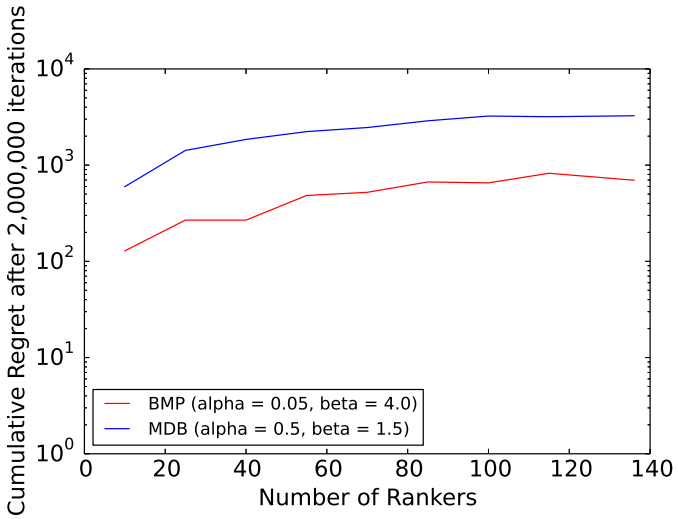


FIGURE 4.9: Cumulative regret after 5,000,000 iterations averaged over 10 runs against the number of rankers being evaluated for the MSLR dataset with navigational click model.

Chapter 5

Conclusions and Future Work

Efficient online evaluation of ranking algorithms is an important problem for large-scale commercial web search engines and recommender systems [112, 70]. Multileaving was introduced in [102] and offers order of magnitudes improvements in efficiency over alternative online evaluation methods. This thesis investigated multileaving and how to manage the exploration-exploitation tradeoff associated with online evaluation using multileaving.

5.1 Multileaving

This thesis introduced two new multileaving algorithms. Furthermore, we demonstrated that an interaction between the two stages of multileaving algorithms can cause substantial inaccuracies if not properly accounted for.

Sample-Only Scored Multileave (SOSM) [17] is the first multileaving method to scale well with the number of rankers being compared. The central idea of SOSM is that rankers are evaluated only on the basis of their relative rankings of the documents included in the multileaved ranking. This means that each recorded click can be used to infer something about the relative quality of each pair of rankers. This is in contrast to prior methods such as TDM where a click was only used to infer something about the quality of a single ranker, or PM, where some rankers had more opportunity to obtain credit than others, independent of their actual quality. This efficient use of feedback allows SOSM to estimate the relative qualities of rankers, even after very few comparisons, regardless of how many rankers are being compared. SOSM was found to be substantially more efficient than prior multileaving methods, with prior methods often requiring twice as many user interactions to obtain similarly good ranker quality estimates [17].

The main non-algorithmic contribution of this thesis is that we demonstrate that the scoring function in a multileaving method must correct for the probability a document has of being shown to the user. Prior multileaving methods, including SOSM, did not account for this, and consequently, and contrary to initial evaluations of multileaving methods, they are prone to being inaccurate.

The second multileaving algorithm contributed as part of this thesis, Multileaving using Importance Sampling (MIS), is the first multileaving algorithm to properly account for the probabilities of documents being presented to the user when scoring rankers. This is therefore the first multileaving method, which can reliably be used to accurately estimate the quality

of rankers. MIS was shown to be highly accurate, scale well with the number of rankers being compared, and was shown to reliably evaluate rankers. Finally, MIS is the first multileaving method that can be used in an unbiased manner on historical interaction data. This is possible because the importance sampling in the scoring function allows MIS to compensate for the difference in probability that a given document was presented to the user in the historical data, and the probability that the document would be presented to the user if a new multileaving was performed.

Thus, this thesis demonstrated (i) that there exist substantial problems with multileaving, in that the scoring function and the document sampling strategy of the multileaving method interact with each other; and (ii) that these problems can be overcome by using importance sampling to correct for this interaction. Consequently, multileaving can provide unbiased, scalable, efficient and accurate evaluations. This suggests that multileaving could have the potential to play an important part in the online evaluation of large-scale web search and recommender systems.

An important caveat regarding these findings is that the experiments for this thesis were carried out on simulated users. More specifically, they used Lerot [100], a framework for simulated experiments in online evaluation and online learning to rank. Although these experiments with simulated users have become the standard in the interleaving and multileaving literature, more work is needed to verify how well the outcomes of these simulated experiments agree with experiments carried out on deployed search systems. Thus, there is a need to evaluate multileaving algorithms on real, deployed search systems. This is needed to verify that biases not captured by the user models from our experiments do not substantially alter the outcomes of experiments. More generally, recalling the complicated interplay between system effectiveness and user satisfaction, there is a need to verify that the preferences inferred by multileaving methods agree with measures of user satisfaction.

There are several interesting possible improvements and extensions to MIS which remain as future work. One promising approach is to investigate the possibility of using weighted importance sampling, and document sampling techniques which preferentially sample from those rankers for which our quality estimates are most uncertain, in order to improve the tradeoff between learning efficiency and quality of the multileaved list. Additionally, if this tradeoff can be handled better, there is the potential to investigate document sampling strategies for which we more strongly control the order of the documents presented during multileaving.

For our algorithm, MIS, we have theoretical guarantees that the expected outcome of evaluation can be made to agree with A/B testing. However, this guarantee applies to a particular class of metrics for A/B testing, and an important avenue for future research would be to extend the work of [99] to multileaving. Here we would want to investigate experimentally what types of A/B metrics can be captured accurately by multileaving approaches. Furthermore, it would be useful to extend the theoretical guarantee offered by MIS to more general classes of A/B metrics.

Current multileaving methods are applicable to ranking algorithms which output ranked lists of documents. Since modern search engines aggregate results from multiple sources, multileaving could be made more widely applicable if work on applying interleaving to aggregated search and more general

user interfaces [66, 29] could be extended to multileaving. Another challenge is to ensure that multileaving does not interfere with desirable properties of search result pages such as diversity.

5.2 Exploration-Exploitation Tradeoff

One of the key drawbacks of online evaluation methods is that the outputs of new, potentially poor, rankers are presented to actual users. If a new ranker is poor, users will be presented with poor results and, in the worst case, might abandon the service [54]. Conversely, if new rankers are not presented there is a risk of overlooking better rankers in the pool of rankers. In online learning the question of determining a proper exploration level is known as the *exploration-exploitation tradeoff*.

This thesis introduced new algorithms for managing the exploration-exploitation tradeoff applicable to two classes of multileaving algorithms. The first, Multi-dueling Bandits (MDB), uses Hoeffding's inequality and handles multileaving algorithms such as SOSM, which output relative outcomes for the rankers being compared. The second, Bandits with Multiple Plays (BMP), uses an empirical Bernstein inequality and handles multileaving algorithms such as MIS, which output absolute scores for the rankers being compared. These algorithms perform similarly, and both give orders of magnitude improvements in performance over state-of-the-art dueling bandit algorithms.

In particular, the performance of both algorithms scales substantially better with the number of rankers being compared than dueling bandit algorithms. This suggests that the MDB and BMP settings could allow for more aggressive online experimentation in terms of the number of new rankers evaluated.

We have not provided any theoretical upper or lower bounds on the performance obtainable in our problem settings. This is an important area for future work, in particular, it would be interesting to see to what extent the scalability of performance is improved in a theoretical sense over the bandits and dueling bandits settings. Subsequent work in multi-dueling bandits [107] has investigated how to model dependencies between arms, and extending this to the bandits with multiple plays setting could provide improved performance, since the scores obtained by rankers when comparing them using MIS are likely to be correlated based on how similar the rankers are.

Work on adversarial bandits has recently been extended to the dueling bandits setting [45]. Here the winning probabilities for different arms are not necessarily identically and independently distributed. The problem of formulating and providing algorithms for adversarial MDB or BMP settings could be an interesting problem. An example of a realistic use case for such algorithms could be the fact that ranker qualities are likely to drift over time in the web search setting, and being able to account for this could avoid the problem of too heavily selecting strong rankers which subsequently become weaker.

In extension to this, recent work has focused on algorithms which are strong for both stochastic and adversarial bandit settings [19, 104, 7, 103]. Extending this work to the dueling bandits, MDB and BMP settings could

allow for algorithms which can exploit the presence of a stochastic structure to the problem, while not performing poorly when this is absent.

Finally, it is likely that a single ranking algorithm may not be optimal for all use cases, and extending work on contextual bandits to the MDB or BMP settings could be important for practical applications. Here the goal is not to focus on a single ranker which is optimal for every situation, but instead to select rankers which are the best for the given context.

5.3 Other Future Work

Online learning to rank approaches based on interleaving and multileaving, such as dueling bandit gradient descent [124] and multileave gradient descent [101, 85] assume that the parameter space is convex. This assumption is known to be violated in practice [124]. Non convex online learning to rank using multileaving could be possible by using multileaving to simultaneously explore many areas of the parameter space. This could also allow the use of more complicated ranking models, since current online learning to rank approaches focus on linear models.

This thesis focuses purely on multileaving methods which attempt to infer something about the quality of ranking algorithms through click feedback. An important challenge could be to attempt to extend these methods to using and combining other types of implicit feedback. For example, what can be inferred about the relative quality of two interleaved rankers from user abandonment, and how can we simultaneously learn from clicks and other signals such as abandonment?

In this thesis we have treated the problem of multileaving, and the problem of dealing with the associated exploration-exploitation tradeoff as two separate problems. It may be preferable to attempt to treat this as a single problem. This could involve developing a multileaving which balances the need to explore documents about which the rankers disagree, and the need to exploit agreement between the rankers as to which documents are strong. This could be preferable, since we might be able to optimise the displayed documents according to criteria such as the potential information gain from a click on the document. The current approach of treating multileaving and managing the exploration-exploitation tradeoff separately leads to potentially suboptimal exploration, since exploration is carried out based on the choices of rankers, instead of the choices of documents.

Finally, recall that the experiments in this thesis all involved simulated users, and that it would be preferable to test algorithms on real users. Living Labs for Information Retrieval provides an online evaluation framework for ranking algorithms [8, 11]. It could be highly useful for academic research in online evaluation methods if a similar framework could be created, or if the Living Labs framework could be extended, to allow online experimentation with online evaluation methods. This could help alleviate the problem that academic research on online evaluation methods is heavily reliant on simulated user feedback.

Bibliography

[1] Ailon, N., Karnin, Z. S., and Joachims, T. Reducing dueling bandits to cardinal bandits. In *ICML* (2014), vol. 32, pp. 856–864.

[2] Al-Maskari, A., Sanderson, M., and Clough, P. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (2007), ACM, pp. 773–774.

[3] Arguello, J., Diaz, F., and Callan, J. Learning to aggregate vertical results into web search results. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), ACM, pp. 201–210.

[4] Audibert, J.-Y., Munos, R., and Szepesvári, C. Tuning bandit algorithms in stochastic environments. In *ALT* (2007), vol. 4754, Springer, pp. 150–165.

[5] Audibert, J.-Y., Munos, R., and Szepesvári, C. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science 410*, 19 (2009), 1876–1902.

[6] Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning 47*, 2-3 (2002), 235–256.

[7] Auer, P., and Chiang, C.-K. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. *arXiv preprint arXiv:1605.08722* (2016).

[8] Azzopardi, L., and Balog, K. Towards a living lab for information retrieval research and development. In *International Conference of the Cross-Language Evaluation Forum for European Languages* (2011), Springer, pp. 26–37.

[9] Azzopardi, L., Järvelin, K., Kamps, J., and Smucker, M. D. Report on the sigir 2010 workshop on the simulation of interaction. In *ACM SIGIR Forum* (2011), vol. 44, ACM, pp. 35–47.

[10] Baeza-Yates, R., Ribeiro-Neto, B., et al. *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[11] Balog, K., Kelly, L., and Schuth, A. Head first: Living labs for ad-hoc search evaluation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014), ACM, pp. 1815–1818.

[12] Balsubramani, A., Karnin, Z., Schapire, R., and Zoghi, M. Instance-dependent regret bounds for dueling bandits. In *Proceedings of The 29th Conference on Learning Theory, COLT* (2016), vol. 2016.

[13] Berry, D. A., and Fristedt, B. *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer, 1985.

[14] Blanco, R., Halpin, H., Herzig, D. M., Mika, P., Pound, J., Thompson, H. S., and Tran Duc, T. Repeatable and reliable search system evaluation using crowdsourcing. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), ACM, pp. 923–932.

[15] Borisov, A., Markov, I., de Rijke, M., and Serdyukov, P. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 531–541.

[16] Bottou, L., Peters, J., Candela, J. Q., Charles, D. X., Chickering, M., Portugaly, E., Ray, D., Simard, P. Y., and Snelson, E. Counterfactual reasoning and learning systems: the example of computational advertising. *Journal of Machine Learning Research 14*, 1 (2013), 3207–3260.

[17] Brost, B., Cox, I. J., Seldin, Y., and Lioma, C. An improved multileaving algorithm for online ranker evaluation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2016), pp. 745–748.

[18] Brost, B., Seldin, Y., Cox, I. J., and Lioma, C. Multi-dueling bandits and their application to online ranker evaluation. In *Proceedings of the 25th International ACM Conference on Information and Knowledge Management* (2016), ACM, pp. 2161–2166.

[19] Bubeck, S., and Slivkins, A. The best of both worlds: Stochastic and adversarial bandits. In *COLT* (2012), pp. 42–1.

[20] Buckley, C., Dimmick, D., Soboroff, I., and Voorhees, E. Bias and the limits of pooling for large collections. *Information retrieval 10*, 6 (2007), 491–508.

[21] Buscher, G., Dengel, A., Biedert, R., and Elst, L. V. Attentive documents: Eye tracking as implicit feedback for information retrieval and beyond. *ACM Transactions on Interactive Intelligent Systems (TiiS) 1*, 2 (2012), 9.

[22] Büttcher, S., Clarke, C. L., and Cormack, G. V. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.

[23] Büttcher, S., Clarke, C. L., Yeung, P. C., and Soboroff, I. Reliable information retrieval evaluation with incomplete and biased judgements. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (2007), ACM, pp. 63–70.

[24] Carmel, D., Halawi, G., Lewin-Eytan, L., Maarek, Y., and Raviv, A. Rank by time or by relevance?: Revisiting email search. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), ACM, pp. 283–292.

[25] Chapelle, O., and Chang, Y. Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge* (2011), pp. 1–24.

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

[27] Chapelle, O., and Zhang, Y. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 1–10.

[28] Chuklin, A., Markov, I., and Rijke, M. d. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services 7*, 3 (2015), 1–115.

[29] Chuklin, A., Schuth, A., Hofmann, K., Serdyukov, P., and De Rijke, M. Evaluating aggregated search using interleaving. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (2013), ACM, pp. 669–678.

[30] Chuklin, A., Schuth, A., Zhou, K., and Rijke, M. D. A comparative analysis of interleaving methods for aggregated search. *ACM Transactions on Information Systems (TOIS) 33*, 2 (2015), 5.

[31] Chuklin, A., Serdyukov, P., and De Rijke, M. Click model-based information retrieval metrics. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* (2013), ACM, pp. 493–502.

[32] Cooper, W. S. A definition of relevance for information retrieval. *Information storage and retrieval 7*, 1 (1971), 19–37.

[33] Craswell, N., Zoeter, O., Taylor, M., and Ramsey, B. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (2008), ACM, pp. 87–94.

[34] Croft, W. B. What do people want from information retrieval. *D-Lib magazine 1*, 5 (1995).

[35] Croft, W. B. Language models for information retrieval. In *Data Engineering, 2003. Proceedings. 19th International Conference on* (2003), IEEE, pp. 3–7.

[36] Croft, W. B., Metzler, D., and Strohmann, T. *Search engines*. Pearson Education, 2010.

[37] Crook, T., Frasca, B., Kohavi, R., and Longbotham, R. Seven pitfalls to avoid when running controlled experiments on the web. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 1105–1114.

[38] Deng, A., Xu, Y., Kohavi, R., and Walker, T. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the sixth ACM international conference on Web search and data mining* (2013), ACM, pp. 123–132.

[39] Dou, Z., Song, R., and Wen, J.-R. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 581–590.

[40] Dou, Z., Song, R., Wen, J.-R., and Yuan, X. Evaluating the effectiveness of personalized web search. *IEEE Transactions on Knowledge and Data Engineering 21*, 8 (2009), 1178–1190.

[41] Drutsa, A., Ufliand, A., and Gusev, G. 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* (2015), ACM, pp. 763–772.

[42] Dudik, M., Hofmann, K., Schapire, R. E., Slivkins, A., and Zoghi, M. Contextual dueling bandits. In *Proceedings of the 28th Conference on Learning Theory* (2015).

[43] Dupret, G. E., and Piwowarski, B. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (2008), ACM, pp. 331–338.

[44] Frøkjær, E., Hertzum, M., and Hornbæk, K. Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (2000), ACM, pp. 345–352.

[45] Gajane, P., Urvoy, T., and Clérot, F. A relative exponential weighing algorithm for adversarial utility-based dueling bandits. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37* (2015), ICML'15, JMLR.org, pp. 218–227.

[46] Granka, L. A., Joachims, T., and Gay, G. Eye-tracking analysis of user behavior in www search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (2004), ACM, pp. 478–479.

[47] Guha, R., McCool, R., and Miller, E. Semantic search. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 700–709.

[48] Guo, F., Liu, C., Kannan, A., Minka, T., Taylor, M., Wang, Y.-M., and Faloutsos, C. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 11–20.

[49] Guo, F., Liu, C., and Wang, Y. M. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (2009), ACM, pp. 124–131.

[50] Harman, D. Overview of the second text retrieval conference (trec-2). *Information Processing & Management 31*, 3 (1995), 271–289.

[51] Hassan, A., Shi, X., Craswell, N., and Ramsey, B. Beyond clicks: query reformulation as a predictor of search satisfaction. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (2013), ACM, pp. 2019–2028.

[52] Hersh, W., Turpin, A., Price, S., Kraemer, D., Olson, D., Chan, B., and Sacherek, L. Challenging conventional assumptions of automated information retrieval with real users: Boolean searching and batch retrieval evaluations. *Information Processing & Management 37*, 3 (2001), 383–402.

[53] Hofmann, K., Li, L., Radlinski, F., et al. Online evaluation for information retrieval. *Foundations and Trends® in Information Retrieval 10*, 1 (2016), 1–117.

[54] Hofmann, K., Whiteson, S., and de Rijke, M. Balancing exploration and exploitation in learning to rank online. In *Advances in Information Retrieval*. Springer, 2011, pp. 251–263.

[55] Hofmann, K., Whiteson, S., and de Rijke, M. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), ACM, pp. 249–258.

[56] Hofmann, K., Whiteson, S., and Rijke, M. D. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS) 31*, 4 (2013), 17.

[57] Huang, J., White, R. W., and Dumais, S. No clicks, no problem: using cursor movements to understand and improve search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), ACM, pp. 1225–1234.

[58] Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), Acm, pp. 154–161.

[59] Jøsang, A., Ismail, R., and Boyd, C. A survey of trust and reputation systems for online service provision. *Decision support systems 43*, 2 (2007), 618–644.

[60] Kanoulas, E. A short survey on online and offline methods for search quality evaluation. In *Information Retrieval*. Springer International Publishing, 2016, pp. 38–87.

[61] Kanoulas, E., and Aslam, J. A. Empirical justification of the gain and discount function for ndcg. In *Proceedings of the 18th ACM conference on Information and knowledge management* (2009), ACM, pp. 611–620.

[62] Kelly, D., et al. Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends® in Information Retrieval 3*, 1–2 (2009), 1–224.

[63] Kelly, D., and Teevan, J. Implicit feedback for inferring user preference: a bibliography. In *ACM SIGIR Forum* (2003), vol. 37, ACM, pp. 18–28.

[64] Keskustalo, H., Järvelin, K., and Pirkola, A. Evaluating the effectiveness of relevance feedback based on a user simulation model: effects of a user scenario on cumulated gain value. *Information Retrieval 11*, 3 (2008), 209–228.

[65] Kharitonov, E., Drutsa, A., and Serdyukov, P. Learning sensitive combinations of a/b test metrics. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017), ACM, pp. 651–659.

[66] Kharitonov, E., Macdonald, C., Serdyukov, P., and Ounis, I. Generalized team draft interleaving. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), ACM, pp. 773–782.

[67] Kharitonov, E., Macdonald, C., Serdyukov, P., and Ounis, I. Optimised scheduling of online experiments. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 453–462.

[68] Kharitonov, E., Vorobev, A., Macdonald, C., Serdyukov, P., and Ounis, I. Sequential testing for early stopping of online experiments. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 473–482.

[69] Kohavi, R., Deng, A., Frasca, B., Longbotham, R., Walker, T., and Xu, Y. Trustworthy online controlled experiments: Five puzzling outcomes explained. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM, pp. 786–794.

[70] Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., and Pohlmann, N. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 1168–1176.

[71] Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery 18*, 1 (2009), 140–181.

[72] Komiyama, J., Honda, J., Kashima, H., and Nakagawa, H. Regret lower bound and optimal algorithm in dueling bandit problem. *arXiv preprint arXiv:1506.02550* (2015).

[73] Komiyama, J., Honda, J., and Nakagawa, H. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. *arXiv preprint arXiv:1506.00779* (2015).

[74] Komiyama, J., Honda, J., and Nakagawa, H. Copeland dueling bandit problem: Regret lower bound, optimal algorithm, and computationally efficient algorithm. *arXiv preprint arXiv:1605.01677* (2016).

[75] Langford, J., Strehl, A., and Wortman, J. Exploration scavenging. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 528–535.

[76] Li, J., Huffman, S., and Tokuda, A. Good abandonment in mobile and pc internet search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (2009), ACM, pp. 43–50.

[77] Li, L., Kim, J. Y., and Zitouni, I. Toward predicting the outcome of an a/b experiment for search relevance. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (2015), ACM, pp. 37–46.

[78] Liu, T.-Y., et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval 3*, 3 (2009), 225–331.

[79] Manning, C. D., Raghavan, P., Schütze, H., et al. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.

[80] Maxwell, D., and Azzopardi, L. Simulating interactive information retrieval: Simiir: A framework for the simulation of interaction. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (2016), ACM, pp. 1141–1144.

[81] Mizzaro, S. How many relevances in information retrieval? *Interacting with computers 10*, 3 (1998), 303–320.

[82] Murdock, V., and Lalmas, M. Workshop on aggregated search. In *ACM SIGIR Forum* (2008), vol. 42, ACM, pp. 80–83.

[83] Nolting, M., and von Seggern, J. E. Context-based a/b test validation. In *Proceedings of the 25th International Conference Companion on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 277–278.

[84] Nowak, S., and Rüger, S. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval* (2010), ACM, pp. 557–566.

[85] Oosterhuis, H., Schuth, A., and de Rijke, M. Probabilistic multileave gradient descent. In *European Conference on Information Retrieval* (2016), Springer, pp. 661–668.

[86] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford InfoLab, 1999.

[87] Pandelis, D. G., and Teneketzis, D. On the optimality of the gittins index rule for multi-armed bandits with multiple plays. *Mathematical Methods of Operations Research 50*, 3 (1999), 449–461.

[88] Park, T. K. The nature of relevance in information retrieval: An empirical study. *The library quarterly 63*, 3 (1993), 318–351.

[89] Radlinski, F., and Craswell, N. Optimized interleaving for online retrieval evaluation. In *Proceedings of the sixth ACM international conference on Web search and data mining* (2013), ACM, pp. 245–254.

[90] Radlinski, F., Kurup, M., and Joachims, T. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), ACM, pp. 43–52.

[91] Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval 3*, 4 (2009), 333–389.

[92] Robertson, S. E. The probability ranking principle in ir. *Journal of documentation 33*, 4 (1977), 294–304.

[93] Salojärvi, J., Kojo, I., Simola, J., and Kaski, S. Can relevance be inferred from eye movements in information retrieval. In *Proceedings of WSOM* (2003), vol. 3, pp. 261–266.

[94] Salton, G. Associative document retrieval techniques using bibliographic information. *Journal of the ACM (JACM) 10*, 4 (1963), 440–457.

[95] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing & management 24*, 5 (1988), 513–523.

[96] Sanderson, M. *Test collection based evaluation of information retrieval systems*. Now Publishers Inc, 2010.

[97] Sanderson, M., Paramita, M. L., Clough, P., and Kanoulas, E. Do user preferences and evaluation measures line up? In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (2010), ACM, pp. 555–562.

[98] Schuth, A., Bruintjes, R.-J., Buüttner, F., van Doorn, J., Groenland, C., Oosterhuis, H., Tran, C.-N., Veeling, B., van der Velde, J., Wechsler, R., et al. Probabilistic multileave for online retrieval evaluation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 955–958.

[99] Schuth, A., Hofmann, K., and Radlinski, F. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 463–472.

[100] Schuth, A., Hofmann, K., Whiteson, S., and de Rijke, M. Lerot: An online learning to rank framework. In *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation* (2013), ACM, pp. 23–26.

[101] Schuth, A., Oosterhuis, H., Whiteson, S., and de Rijke, M. Multileave gradient descent for fast online learning to rank. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (2016), ACM, pp. 457–466.

[102] Schuth, A., Sietsma, F., Whiteson, S., Lefortier, D., and de Rijke, M. Multileaved comparisons for fast online evaluation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014), ACM, pp. 71–80.

[103] Seldin, Y., and Lugosi, G. An improved parametrization and analysis of the exp3++ algorithm for stochastic and adversarial bandits. *arXiv preprint arXiv:1702.06103* (2017).

[104] Seldin, Y., and Slivkins, A. One practical algorithm for both stochastic and adversarial bandits. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014* (2014), pp. 1287–1295.

[105] Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation 28*, 1 (1972), 11–21.

[106] Sui, Y., and Burdick, J. Clinical online recommendation with subgroup rank feedback. In *Proceedings of the 8th ACM Conference on Recommender systems* (2014), ACM, pp. 289–292.

[107] Sui, Y., Zhuang, V., Burdick, J. W., and Yue, Y. Multi-dueling bandits with dependent arms. *arXiv preprint arXiv:1705.00253* (2017).

[108] Swaminathan, A., and Joachims, T. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research 16* (2015), 1731–1755.

[109] Swaminathan, A., and Joachims, T. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proceedings of the 32nd International Conference on Machine Learning* (2015), pp. 814–823.

[110] Swaminathan, A., and Joachims, T. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems* (2015), pp. 3231–3239.

[111] Swaminathan, A., Krishnamurthy, A., Agarwal, A., Dudík, M., Langford, J., Jose, D., and Zitouni, I. Off-policy evaluation for slate recommendation. *arXiv preprint arXiv:1605.04812* (2016).

[112] Tang, D., Agarwal, A., O'Brien, D., and Meyer, M. Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), ACM, pp. 17–26.

[113] Teevan, J., Dumais, S. T., and Horvitz, E. Potential for personalization. *ACM Transactions on Computer-Human Interaction (TOCHI) 17*, 1 (2010), 4.

[114] Turney, P. D., and Pantel, P. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research 37* (2010), 141–188.

[115] Turtle, H. Natural language vs. boolean query evaluation: A comparison of retrieval performance. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (1994), Springer-Verlag New York, Inc., pp. 212–220.

[116] Uchiya, T., Nakamura, A., and Kudo, M. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory* (2010), Springer, pp. 375–389.

[117] Urvoy, T., Clerot, F., Féraud, R., and Naamane, S. Generic exploration and k-armed voting bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (2013), pp. 91–99.

[118] Voorhees, E. M. The philosophy of information retrieval evaluation. In *Workshop of the Cross-Language Evaluation Forum for European Languages* (2001), Springer, pp. 355–370.

[119] Wang, X., Bendersky, M., Metzler, D., and Najork, M. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (2016), ACM, pp. 115–124.

[120] Wang, Y., Wang, L., Li, Y., He, D., Chen, W., and Liu, T.-Y. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)* (2013).

[121] Wasserman, L. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[122] Wu, H., and Liu, X. Double thompson sampling for dueling bandits. In *Advances in Neural Information Processing Systems* (2016), pp. 649–657.

[123] Yue, Y., Broder, J., Kleinberg, R., and Joachims, T. The k-armed dueling bandits problem. *Journal of Computer and System Sciences 78*, 5 (2012), 1538–1556.

[124] Yue, Y., and Joachims, T. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 1201–1208.

[125] Yue, Y., and Joachims, T. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (2011), pp. 241–248.

[126] Yue, Y., Patel, R., and Roehrig, H. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 1011–1018.

[127] Zhai, C., and Lafferty, J. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), ACM, pp. 334–342.

[128] Zhou, K., Cummins, R., Lalmas, M., and Jose, J. M. Evaluating aggregated search pages. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (2012), ACM, pp. 115–124.

[129] Zobel, J. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), ACM, pp. 307–314.

[130] Zoghi, M., Karnin, Z. S., Whiteson, S., and De Rijke, M. Copeland dueling bandits. In *Advances in Neural Information Processing Systems* (2015), pp. 307–315.

[131] Zoghi, M., Whiteson, S., and de Rijke, M. Mergerucb: A method for large-scale online ranker evaluation. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (2015), ACM, pp. 17–26.

[132] Zoghi, M., Whiteson, S., Munos, R., Rijke, M. d., et al. Relative upper confidence bound for the k-armed dueling bandit problem. In *JMLR Workshop and Conference Proceedings* (2014), JMLR, pp. 10–18.

[133] Zoghi, M., Whiteson, S. A., de Rijke, M., and Munos, R. Relative confidence sampling for efficient on-line ranker evaluation. In *Proceedings of the 7th ACM international conference on Web search and data mining* (2014), ACM, pp. 73–82.