Fast and Reliable Online Learning to Rank for Information Retrieval

Katja Hofmann

Fast and Reliable Online Learning to Rank for Information Retrieval

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op dinsdag 28 mei 2013, te 12:00 uur

door

Katja Hofmann

geboren te Borna, Duitsland

SIKS Dissertation Series No. 2013-20
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.061.814 and a Google Anita Borg Memorial Scholarship.

ISBN: 978-94-6182-270-3

Acknowledgements

Now that I am writing this section of my thesis, it feels like a luxury to be able to take the time and look back at the last few years, and think of all the people that have affected me throughout this time. So many people have affected me in so many ways that it is impossible to name them all; I will do my best to name all those that most strongly impacted this journey.

I started working on my PhD project four and a half years ago. At that time I had already been working in the Information and Language Processing Systems (ILPS) group at the University of Amsterdam for two years, and very much enjoyed working there. This strongly affected my decision to spend the next few years here. Central to this group is my supervisor, Maarten de Rijke, and he is the person I have to thank most. I am deeply grateful for his guidance and support throughout these years. He always had an open ear for me, despite being one of the busiest and most productive people I know. Most importantly, Maarten has created a research group that is built on trust and mutual respect, and that is a supportive environment for learning, cooperation, and discussion.

ILPS is part of the Intelligent Systems Lab Amsterdam. Working with colleagues from research areas other than information retrieval has taught me to think more broadly and look beyond the horizon of one particular area. The person from ISLA who most strongly affected me is Shimon Whiteson, who eventually became my co-promoter. The first sketches of what was to become the core of my thesis were developed over the course of several long discussions, sitting in the summer sun outside "our old building", where we tried to combine information retrieval and reinforcement learning. I am particularly grateful for the discussions we had throughout the past years. I thank Shimon for challenging me, by asking exactly the right questions and not accepting easy answers.

I am very honored and grateful to have Nick Craswell, Lynda Hardman, Thorsten Joachims, Arjen de Vries, and Max Welling serving as my committee members.

I thank all colleagues at ILPS and ISLA, for the reading groups and discussions, quick chats at the coffee machines, and for creating a work atmosphere that I always enjoyed and where going to work was never a chore. Some colleagues turned into dear friends, and I particularly thank Loredana, Valentin, Bouke, Jiyin, Manos, Wouter, Edgar, Marc and Simon for all the evenings at De Polder, winter vacations, hiking trips, bike excursions, dinners, movie nights, concerts and parties, and all the cherished memories of these. With many colleagues I collaborated on projects and publications, and I thank especially Anne, Erik, Martha, and Krisztian, and my co-authors from outside of ILPS, Toine Bogers, Fritz Behr, Filip Radlinski, Evgeni Tsivtsivadze, and Tom Heskes for all that I have learned in the process.

Two colleagues and friends that have accompanied me throughout my journey towards a PhD are Jiyin and Manos, and I am very happy that they have agreed to be my paranymphs. My thanks also go to Anne, Isaac, Aleksander, and Wouter, for reading through parts of my thesis and providing valuable feedback, and to Mark for designing the cover of my thesis. I thank Joost Kircz for his support, especially towards the beginning of my PhD.

I thank my students for their curiosity, trust and enthusiasm. I am indebted to Auke and Jeroen for setting up and maintaining the computing infrastructure for ILPS. We are

very lucky to have you! I thank our support staff for making sure that our group and our institute run as smoothly as possible every day, especially Virginie, Saskia, and Petra.

Before joining ILPS, I tried my first steps in doing research with my mentor at California State University, Hilary Holz. I thank Hilary Holz, in particular for convincing me to pursue a PhD.

I thank the Netherlands Organisation for Scientific Research (NWO) for sponsoring my PhD project. I thank Google and the Dutch Research School for Information and Knowledge Systems (SIKS) for their additional support. Part of the experiments I report on in this thesis were run on the Lisa Compute Cluster, and I thank SURFsara Computing and Networking Services (`www.surfsara.nl`) for their support in using this cluster.

While working towards my PhD, I had the opportunity to do two internships, one at Google, U.K. and one at Microsoft Bing / Microsoft Research. I am grateful for these opportunities, and I thank my mentors and colleagues there for welcoming me and making me feel part of the team so quickly. I thank them for the exciting projects I got to work on and for sharing their experience.

I very much enjoyed becoming part of a global research community over the course of the past years. It was fun to meet PhD students and other international colleagues at workshops and conferences, and I am grateful for the many things they have taught me. I thank the anonymous reviewers of my papers. We may not always agree, but my work has always improved due to your feedback.

De afgelopen jaren waren een intellectuele reis, maar ook in de rest van mijn leven is er van alles veranderd. Mijn huisgenoten op de Herengracht hebben mijn successen en tegenslagen van dichtbij meegemaakt. Door hen is een huis een thuis geworden waar ik altijd graag naartoe kwam en kom. Van hen heb ik van alles geleerd, over Nederland en de wereld, over politiek, kunst en cultuur, wetenschap en filosofie, en over het leven. Mijn dank gaat aan Susanne, Hanna, Hilde, Charlotte, Daniël, Tomás, Sebastian, en Valentijn.

Ook Jasper heeft ervoor gezorgd dat ik de afgelopen jaren niet alleen maar aan mijn werk dacht. Dank je!

Mein Dank geht an meine Familie, deren Unterstützung ich mir immer sicher sein konnte. Besonders dankbar bin ich meinen Eltern, Angelika und Dietmar. Sie gaben mir Neugier und Erkundungsdrang, und Selbstbewusstsein. Ich danke meinem Bruder Stefan für die Erfahrungen und Gespräche die nur möglich sind wenn man jemanden schon sein Leben lang kennt. Ich danke meiner Oma, Gertrud, für ihre Kraft, und dafür dass sie all meine Telefonnummern auswendig lernt, wohin ich auch gehe.

Katja Hofmann
Amsterdam, 13 April 2013

Contents

1 Introduction

The goal of the research area of information retrieval (IR) is to develop the insights and technology needed to provide access to data collections. The most prominent applications, web search engines like Bing, Google, or Yahoo!, provide instant and easy access to vast and constantly growing collections of web pages. A user looking for information submits a query to the search engine, receives a ranked list of results, and follows links to the most promising ones. To address the flood of data available on the web, today's web search engines have developed into very complex systems. They combine hundreds of ranking features (properties of the query, a document, and the relationship between the two) with the goal of creating the best possible search results for all their users at all times.[1]

A *ranker* (or ranking function) is the part of a search engine that determines the order in which documents retrieved for a given user query should be presented. Until recently, most rankers were developed manually, based on expert knowledge. Developing a good ranker may be easy for some search tasks, but in many cases what constitutes a good ranking depends on the search context, such as users' background knowledge, age, or location, or their specific search goals and intents (Besser et al., 2010; Hofmann et al., 2010a; Rose and Levinson, 2004; Shen et al., 2005). And even though there is an enormous variety in the tasks and goals encountered in web search, web search engines are only the tip of the iceberg. More specialized systems are everywhere: search engines for companies' intranets, local and national libraries, universities' course catalogues, and users' personal documents (e.g., photos, emails, and music) all provide access to different, more or less specialized, document collections, and cater to different users with different search goals and expectations. Addressing each of these settings manually is infeasible. Instead, we need to look for scalable methods that can learn good rankings without expensive, and necessarily limited, manual or semi-manual tuning.

For automatically tuning the parameters of a ranking function, machine learning algorithms are invaluable (Liu, 2009). Most methods employ *supervised learning to rank*, i.e., algorithms are trained on examples of relevant and non-relevant documents for particular queries. Data to train these approaches is typically obtained from experts who label document-query pairs, which is time-consuming and expensive. In many settings, such as personalized or localized search, or when deploying a search engine for a com-

[1] `http://www.wired.com/magazine/2010/02/ff_google_algorithm/all/`, retrieved December 29, 2012.

pany's intranet or a library catalogue, collecting the large amounts of training data required for supervised learning is usually not feasible (Sanderson, 2010). Even in environments where training data is available, it may not capture typical information needs and user preferences perfectly (Radlinski and Craswell, 2010), and cannot anticipate future changes in user needs.

In this thesis we follow an alternative approach, called *online learning to rank*. This technology can enable "self-learning search engines" that learn directly from natural interactions with their users. Such systems promise to be able to continuously adapt and improve their rankings to the specific setting they are deployed in, and continue to learn for as long as they are being used.

Learning directly from user interactions is fundamentally different from the currently dominant supervised learning to rank approaches for IR, where training data is assumed to be randomly sampled from some underlying distribution, and where absolute and reliable labels are provided by professional annotators. In an online learning setting, feedback for learning is a by-product of natural user interactions. This strongly affects what kind of feedback can be obtained, and the quality of the obtained feedback. For example, users expect to be presented with useful results at all times, so trying out new rankings (called *exploration*) can have a high cost in user satisfaction and needs to be balanced against possible future learning gains. Also, feedback inferred from user interactions can be noisy, and it may be affected by how search results are presented (one example of such an effect is *caption bias*). Learning from such lower-quality feedback may result in degraded learning, unless we can design learning to rank algorithms that are robust against these effects. In this thesis we investigate the principles that allow effective online learning to rank for IR, and translate our insights into new algorithms for fast and reliable online learning.

1.1 Research Outline and Questions

The broad question that motivates the research for this thesis is: *Can we build search engines that automatically learn good ranking functions by interacting with their users?* Individual components towards solving this problem already exist (see Chapter 2 for an overview), but other aspects, such as how to learn from noisy and relative feedback, have not yet been investigated. This thesis aims to close some of these gaps, contributing to the long-term goal of a complete online learning to rank solution for IR.

We start our investigation by focusing on the type and quality of feedback that can be obtained for learning to rank in an online setting. Extracting reliable and useful feedback for learning to rank from natural user interactions is difficult, because user interactions are noisy and context-dependent. The most effective techniques identified so far focus on extracting relative information, i.e., they infer user preferences between documents or whole result rankings. In this thesis we focus on these relative feedback techniques, and particularly on so-called *interleaved comparison* methods that infer preferences between rankings using click data. Besides in online learning to rank applications, these methods are used for *online evaluation* for search engine research and development in general.

Given that three interleaved comparison methods have been developed previously, we first aim to understand how these methods compare to each other, i.e., how can we decide which method to use for an online learning to rank system, or to evaluate a given retrieval system? We formalize criteria for analyzing interleaved comparison methods, to answer the following questions:

RQ 1 What criteria should an interleaved comparison method satisfy to enable reliable online learning to rank for IR?

RQ 2 Do current interleaved comparison methods satisfy these criteria?

In answering these questions, we identify three minimal criteria that interleaved comparison methods should satisfy: fidelity, soundness, and efficiency. An interleaved comparison method has fidelity if the quantity it measures, i.e., the expected outcome of ranker comparisons, properly corresponds to the true relevance of the ranked documents. It is sound if its estimates of that quantity are statistically sound, i.e., unbiased and consistent. It is efficient if those estimates are accurate with only little data.

Analyzing previously developed interleaved comparison methods, we find that none of them exhibit fidelity. To address this shortcoming, we develop a new interleaved comparison method, probabilistic interleave (PI), that is based on a probabilistic interpretation of the interleaving process. An extension of PI, PI-MA, is then derived to increase the method's efficiency by marginalizing over known variables instead of using noisy estimates. Regarding these new methods, we address the following questions:

RQ 3 Do PI and its extension PI-MA exhibit fidelity and soundness?

RQ 4 Is PI-MA more efficient than previous interleaved comparison methods? Is it more efficient than PI?

While previous interleaved comparison methods required collecting new data for each ranker comparison, our probabilistic framework enables the reuse of previously collected data. Intuitively, the information contained in these previously collected lists and user clicks should provide some information about the relative quality of new target rankers. However, the source distribution under which the data was collected may differ from the target distribution under which samples would be collected if the new target rankers were compared with live data. This can result in biased estimates of comparison outcomes. To address this problem, we design a second extension of PI, PI-MA-IS. It uses *importance sampling* to compensate for differences between the source and target distribution, and marginalization to maintain high efficiency. Investigating this method analytically and experimentally allows us to address the following questions:

RQ 5 Can historical data be reused to compare new ranker pairs?

RQ 6 Does PI-MA-IS maintain fidelity and soundness?

RQ 7 Can PI-MA-IS reuse historical data effectively?

We then turn to more practical issues of using interleaved comparisons in a web search setting. In this setting, user clicks may be affected by aspects of result pages other than true result relevance, such as how results are presented. If such visual aspects affect user clicks, the question becomes what click-based evaluation really measures. We address the following questions:

RQ 8 (How) does result presentation affect user clicks (caption bias)?

RQ 9 Can we model caption bias, and compensate for it in interleaving experiments?

RQ 10 (How) does caption bias affect interleaving experiments?

After addressing the above questions regarding the quality of feedback that can be obtained in online learning to rank for IR settings, we turn to the principles of online learning to rank for IR. Given the characteristics and restrictions of online learning to rank, we investigate how to perform as effectively as possible in this setting. One central challenge that we formulate is the *exploration-exploitation dilemma*. In an online setting, a search engine learns continuously, while interacting with its users. To satisfy users' expectations as well as possible at any point in time, the system needs to *exploit* what it has learned up to this point. It also needs to *explore* possibly better solutions, to ensure continued learning and improved performance in the future. We hypothesize that addressing this dilemma by balancing exploration and exploitation can improve online performance. We design two algorithms for achieving such a balance in pairwise and listwise online learning to rank:

RQ 11 Can balancing exploration and exploitation improve online performance in online learning to rank for IR?

RQ 12 How are exploration and exploitation affected by noise in user feedback?

RQ 13 How does the online performance of different types (pairwise and listwise) of online learning to rank for IR approaches relate to balancing exploration and exploitation?

Finally, we return to the question of how to learn as quickly and effectively as possible in an online learning to rank for IR setting. We hypothesize that reusing click data that was collected during earlier learning steps could be used to gain additional information about the relative quality of new rankers. Based on our PI-MA-IS method for reusing historical data, we develop two algorithms for learning with historical data reuse. The first, RHC, reuses historical data to make ranker comparisons during learning more reliable. The other, CPS, uses historical data for more effective exploration of the solution space. The research questions addressed by our subsequent research are:

RQ 14 Can previously observed (historical) interaction data be used to speed up online learning to rank?

RQ 15 Is historical data more effective when used to make comparisons more reliable (as in RHC), or when used to increase local exploration (as in CPS)?

RQ 16 How does noise in user feedback affect the reuse of historical interaction data for online learning to rank?

1.2 Main Contributions

In this section we summarize the main algorithmic, theoretical, and empirical contributions of this thesis.

Algorithmic contributions:

- A probabilistic interleaved comparison method, called probabilistic interleave (PI), that exhibits fidelity, and an extension of PI, called PI-MA, that increases the efficiency of PI by marginalizing over known variables instead of using noisy observations.
- The first interleaved comparison method that allows reuse of historical interaction data (called PI-MA-IS, an extension of PI-MA).
- An approach for integrating models of caption bias with interleaved comparison methods in order to compensate for caption bias in interleaving experiments.
- The first two online learning to rank for IR algorithms (one pairwise, one listwise approach) that can balance exploration and exploitation.
- The first two online learning to rank algorithms that can utilize previously observed (historical) interaction data: reliable historical comparisons (RHC), and candidate preselection (CPS).

Theoretical contributions:

- A framework for analyzing interleaved comparison methods in terms of fidelity, soundness, and (previously proposed) efficiency.
- Analysis of the interleaved comparison methods balanced interleave, team draft, and document constraints, showing that none exhibits fidelity.
- Two proofs that show that our proposed extensions of PI, PI-MA and PI-MA-IS maintain soundness.
- A general-purpose probabilistic model of caption bias in user click behavior that can combine document-pairwise and pointwise features.
- A formalization of online learning to rank for IR as a contextual bandit problem, and formulation of the exploration-exploitation dilemma in this setting.

Empirical contributions:

- An experimental framework that allows for the assessment of online evaluation and online learning to rank methods using annotated learning to rank data sets and click models in terms of their online performance.
- An empirical evaluation of PI, PI-MA, and all existing interleaved comparison methods, showing that PI-MA is the most efficient method.
- An empirical evaluation of interleaved comparison methods under historical data, showing that PI-MA-IS is the only method that can effectively reuse historical data.
- A large-scale assessment of our caption-bias model with pairwise and pointwise feature sets using real click data in a web search setting, showing that pointwise features are most important, but combinations of pointwise and pairwise features are most accurate for modeling caption bias.

- Results of applying caption bias models to interleaving experiments in a web search setting, indicating that caption bias can affect interleaved comparison outcomes.
- The first empirical evidence that shows that balancing exploration and exploitation in online learning to rank for IR can significantly improve online performance in online learning to rank for IR.
- First empirical evidence showing that reusing historical data for online learning to rank can substantially and significantly improve online performance.

In addition to the contributions listed above, the software developed for running online learning to rank experiments following our experimental setup is made freely available (Appendix A). This software package includes reference implementations of the developed interleaved comparison methods (PI, PI-MA, and PI-MA-IS) and online learning to rank approaches (balancing exploration and exploitation in pairwise and listwise online learning, online learning to rank with historical data reuse).

1.3 Thesis Overview

This section gives an overview of the content of each chapter of this thesis. The next chapter (Chapter 2) introduces background for all subsequent chapters. Chapter 3 details the problem formulation used throughout this thesis, and introduces the experimental setup that forms the basis of the empirical evaluations in Chapters 4, 6, and 7.

The next four chapters are the main research chapters of this thesis. Each focuses on a specific aspect of online learning to rank for IR. We start with the most central component, the feedback mechanism, in Chapter 4. This chapter develops a framework for analyzing interleaved comparison methods, and proposes a new, probabilistic, interleaved comparison method (PI) and two extensions for more efficient comparisons (PI-MA), and for comparisons with reuse of historical interaction data (PI-MA-IS). Chapter 5 investigates interleaved comparison methods in a web search setting, and develops models for compensating for caption bias in interleaved comparisons. Chapter 6 focuses on approaches for online learning, and investigates how exploration and exploitation can be balanced in online learning to rank for IR, and whether such a balance can improve the online performance of such systems. Finally, Chapter 7 integrates the interleaved comparison methods developed in Chapter 4 with an online learning to rank algorithm to investigate whether and in what way historical data reuse can speed up online learning to rank for IR. We draw conclusions and give an outlook on future work in Chapter 8.

All research chapters build on background introduced in Chapter 2, and all but Chapter 5 use the experimental setup detailed in Chapter 3. Although ideas developed in earlier chapters are referred to in later chapters, each chapter is relatively self-contained (assuming knowledge of the background material provided in Chapters 2 and 3). An exception is Chapter 7, which builds on the interleaved comparison methods developed in Chapter 4.

Readers familiar with existing online evaluation and online learning to rank approaches can skip over Chapter 2. Also, for readers primarily interested in the main ideas and theoretical contributions of this thesis, it is recommended to skip Chapter 3 and only revisit it to understand empirical results as needed.

1.4 Origins

The following publications form the basis of chapters in this thesis.

- Chapter 4 is based on (Hofmann et al., 2011c, 2012b, 2013c).
- Chapter 5 is based on (Hofmann et al., 2012a).
- Chapter 6 is based on (Hofmann et al., 2011a, 2013b).
- Chapter 7 is based on (Hofmann et al., 2013a).

In addition, Chapter 3 combines material from (Hofmann et al., 2011a,c, 2012b, 2013a,b,c). Finally, this thesis draws from insights and experiences gained in (Besser et al., 2010; Hofmann et al., 2008, 2009a,b, 2010a,b, 2011b; Lubell-Doughtie and Hofmann, 2012; Tsivtsivadze et al., 2012).

2 Background

In this chapter we introduce the concepts and previous work on which this thesis is based. Most immediately relevant are the baseline approaches for online learning to rank, which are presented in §2.5. However, we build up related material in several steps. First, we start with general concepts and terminology from IR (§2.1), and learning to rank for IR (§2.2). Because online learning to rank for IR relies on implicit feedback, such as click data, we review existing work on uses of such data for improving IR systems (in §2.3). In this section, we also detail the feedback mechanisms we build on, namely the interleaved comparison methods *balanced interleave* (BI), *team draft* (TD), and *document constraints* (DC).

Besides IR, this thesis draws on concepts and techniques developed in reinforcement learning (RL), a machine learning paradigm where systems learn from interactions with their environment. Many ideas developed within RL are applicable to online learning to rank for IR, and we review relevant research in this area in §2.4. Finally, we detail the two online learning approaches that form our baseline learning algorithms in §2.5.

2.1 Information Retrieval

The term "information retrieval" was coined only in 1950 (as recalled by Mooers (1960)), but research in this area has been actively pursued for at least the last 100 years. Developed at the end of the 19th and the beginning of the 20th centuries, the first automatic retrieval systems used mechanical solutions to speed up lookup in library catalogues (Sanderson and Croft, 2012). Research in this area accelerated with the technological developments of the following decades – systems based on microfilm were succeeded by punchcards and early computerized systems – and ambitious visions were formulated of systems that would allow users to pave trails through information landscapes (Bush, 1945), and that could mine users' search behavior to learn the language to describe documents from a user's perspective (Mooers, 1960).

Since then, IR has undergone dramatic changes, not least because of the pervasiveness and scale at which recorded information has become widely accessible. However, some of the components and concepts central to IR were first developed during early IR research. In this section, we give a brief overview of the concepts that are central to IR and that we refer to throughout this thesis.

An IR system provides its users with access to information, which is typically stored in the form of a document collection. Interaction between a user and a retrieval system is initiated by the user, with the goal of satisfying some (more or less explicit) information need (Belkin et al., 1982). The user expresses this information need in a query (e.g., as a sequence of keywords, but other forms are possible[1]), and submits the query to the retrieval system. Based on the query, the system's task is to select information to present to the user that is likely to be relevant to the users' information need (the concept of relevance is central to IR, but is notoriously difficult to define; we discuss this concept in detail towards the end of this section). Most often, this takes the form of selecting documents from its collection. The result presentation is typically in the form of a ranking, in the order of probability of the documents' relevance to the user's information need (Robertson, 1977).

A major focus of IR research is the development of retrieval models that capture the relationship between a query and a document. In early retrieval systems, the boolean model was dominant (Salton et al., 1983). This model allows users to formulate queries in the form of logical clauses, and retrieve the set of documents that match the query. Limitations of this model led to the development of weighting schemes that allowed users or systems to assign weights to individual terms to indicate their importance (Salton et al., 1975). A result of this effort was the vector space model (VSM), in which queries and documents are represented by vectors in some space of terms (Salton, 1979). Instead of returning sets of matching documents, retrieval systems based on the VSM return ranked lists in which documents are ordered by their similarity to the user query in vector space. One of the most influential weighting schemes developed for the VSM is *TF-IDF*. In it, terms in document vectors are weighted by their term frequency (TF – the number of times the term occurs in the document) times their inverse document frequency (IDF – the inverse of the number of documents in the collection in which the term occurs).

The concept of document rankings was further formalized in the Probability Ranking Principle, which states that retrieval performance is optimized when systems rank documents by their probability of relevance (Robertson, 1977) (assuming an individual user and independence between documents). Consequently, probabilistic approaches to IR were developed, with BM25 as one of the most widely-known variants (Spärck Jones et al., 2000). The most recently developed major IR approaches are based on statistical language modeling, where documents are modeled as sequences of words that are drawn from an underlying distribution. There, scoring a document for a given query is implemented as estimating the probability that the query and the document are sampled from the same distribution (Hiemstra, 1998; Ponte and Croft, 1998). Besides these major retrieval models, many alternatives and extensions have been proposed. Recent developments include extended language models that take, e.g., term proximity into account (Metzler, 2011) and models based on quantum theory (Van Rijsbergen, 2004).

The developed content-based retrieval models were often found to be effective for finding documents that matched the topic of a query. However, early results of IR re-

[1]The empirical research described in this thesis is conducted on data collections that represent textual documents and text-based queries. However, the developed technology is independent of the type of collection and can, in principle, be applied to any feature-based representation of query-document pairs, where both "query" and "document" are interpreted very broadly (e.g., documents can be any retrievable entity). Our methods require that results for a query can be interleaved (cf., §2.3.1).

search showed that such topical matches are not always sufficient, and that the use of additional sources of information could improve results (Salton, 1963). One information source that has been explored extensively are references or citations that link documents such as scientific papers. Garfield (1964) first proposed to index citations and use them for scientific literature search. A crucial insight was that citation information could be represented as a graph structure that could be analyzed to identify e.g., groups of related documents (Salton, 1963). With the advent of the web, such graph-based methods were extended to authoritative web pages based on the link structure of the web. Key developments, such as the HITS algorithm developed by Kleinberg (1999), and the related PageRank algorithm, proved crucial for effective web search (Brin and Page, 1998). Today, graph-based approaches are extended e.g., to understand community structures on the social web, and are applied to develop tools for social and personalized search (Carmel et al., 2009; Efron, 2011). Besides the large-scale graph structure exploited by algorithms such as PageRank, more fine-grained structural information has been shown to be useful for retrieval (Hofmann et al., 2009b). Extensions of topical retrieval models that take document structure into account are BM25F ("fielded" BM25) (Robertson et al., 2004) and the Indri search engine that is based on a combination of language modeling and inference networks (Metzler and Croft, 2004).

In recent years, a wide variety of additional *contextual factors* have been integrated with retrieval models. For example, terms extracted from users' previous queries and previously visited web pages can capture aspects of users' general interests and cognitive background (Matthijs and Radlinski, 2011; Shen et al., 2005). Similarly, detecting users' search goals and intents can be used to improve retrieval performance (Besser et al., 2010; Teevan et al., 2008). In a study of contextual factors in expert finding, we found that several task-dependent factors, such as media experience, organizational structure, and position of an expert in an organization, could improve retrieval performance (Hofmann et al., 2008, 2010a). Finally, the use of location (Bennett et al., 2011) and temporal information (Berberich et al., 2010) was shown to improve search results in, e.g., web and news search.

Newly developed IR models are evaluated following the strong tradition of empirical research in this area. The Cranfield paradigm, which forms the foundation of the Text Retrieval Conference (TREC — the largest IR evaluation campaign) (Voorhees, 2002) allowed rapid progress toward effective topical retrieval models by abstracting away differences between individual users. This setup concentrates on the basic elements of IR evaluation. Given document-query pairs, expert annotators (also called relevance judges) are required to manually provide relevance judgments, i.e., to annotate whether or in how far a document is considered relevant for a given query (Voorhees and Harman, 2005). Potential differences between judges and other non-topical aspects of relevance were not considered initially. However, extensions of the Cranfield paradigm address interactive IR (Over, 2001), the retrieval of varied information objects (e.g., people, entities, user generated content (Bailey et al., 2007; Balog et al., 2011; Ounis et al., 2008)), and consider relationships between individual documents (in the novelty and diversity tracks (Clarke et al., 2009; Soboroff and Harman, 2003)) and queries (in the interactive and session tracks (Kanoulas et al., 2010; Over, 2001)). Detailed surveys of evaluation in IR and interactive IR can be found in (Sanderson, 2010) and (Kelly, 2009).

Given a TREC-style document collection with relevance judgments, the quality of an

IR system is computed using one or more IR evaluation measures. Early work focused on recall and precision, but many alternatives have been proposed since (Sanderson, 2010). A measure that was developed during the early years of TREC, and that continues to be influential, is Mean Average Precision (MAP), which captures ranking performance in a single summary statistic. Other metrics have been developed to allow for graded relevance judgments (Järvelin and Kekäläinen, 2002), assess the quality of diversified result lists (Clarke et al., 2011), or explicitly model assumptions about user behavior (Chapelle et al., 2009; Yilmaz et al., 2010).

In this thesis we evaluate ranking performance in terms of Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002). This measure was proposed to deal with graded relevance judgments, and is the most commonly used evaluation measure for assessing interleaved comparison methods (Radlinski and Craswell, 2010) and online learning to rank (Yue and Joachims, 2009). We use the formulation from (Burges et al., 2005):[2]

$$NDCG = \sum_{i=1}^{len(\mathbf{l})} \frac{2^{rel(\mathbf{l}[i])} - 1}{\log_2(i+1)} iNDCG^{-1}. \tag{2.1}$$

For a given result list $\mathbf{l}$ of length $len(\mathbf{l})$, this metric sums over the gain that is based on the relevance label $(rel(\mathbf{l}[i]))$ of each document, and divides it by a discount factor (based on the log of the rank i at which the document was presented). This sum is then normalized by the ideal NDCG ($iNDCG$) that would be obtained on an ideal document ranking.

As mentioned above, the goal of retrieval models and evaluation efforts is to correctly select or rank "relevant" documents. Despite being the most central concept of IR research, the meaning of "relevance" has been debated throughout the development of the field. Relevance has been operationalized in many different ways, ranging from *topical* relevance (whether a document is about a given topic), to *cognitive* (concerning the relation between the presented information and a user's cognitive state, e.g., background or domain knowledge) (Ingwersen and Järvelin, 2005) and *situational* (concerning the relation between the presented information and a user's situation) views (Saracevic, 2007). Defining this concept is an interdisciplinary effort, and forms an important overlap between IR and Information Seeking – a research area where the information seeking process itself is the main focus of investigation. Discussions range from aspects of human psychology, where information seeking can be characterized as behavior with the goal of reducing uncertainty (and relevant information is information that indeed reduces uncertainty) (Morrison, 1993; Wilson et al., 2002), to epistemological reflections that characterize relevance in terms of subject knowledge (Hjørland, 2010). Here, we adopt a working definition of relevance as situational. We consider information relevant when it addresses an (implicit or explicit) information need of the user (of an IR system) at a given time and place. In addition, relevance can be graded, i.e., pieces of information can be more or less relevant to a user in a given situation.

The concept of relevance is important to this thesis, because its central motivation is the question of how to design retrieval systems that can address situational relevance. In recent years, the amount and variety of digitally available information has increased

[2]Note that our formulation differs from earlier ones, including the one provided with the LETOR toolkit, where documents at rank 2 are not discounted (cf., (Järvelin and Kekäläinen, 2002)). Here, we use the formulation from (Burges et al., 2005) so that relevance differences at the highest ranks can be detected.

dramatically, as have the types of information needs that more and more heterogenous users try to answer using these systems. These changes have fostered increasing interest in research on contextual factors for improving retrieval systems. This work can be seen as an effort towards addressing situational relevance.

As more contextual factors are identified, we think that different combinations of these factors will be needed to provide optimal results to each user at each point in time. Developing such combinations manually is impossible, so methods for automatically learning such combinations are needed. One solution for automatically tuning retrieval systems is learning to rank for IR. Existing learning to rank methods will be surveyed in the next section. In this thesis, we specifically focus on methods for *online* learning to rank for IR, which can automatically improve a retrieval system based on observed user behavior. This technology enables search engines that interactively adapt to their users, moving closer to the goal of presenting the best possible search results to each user at each point in time.

2.2 Learning to Rank for IR

Current web search systems take many (possibly hundreds) of ranking features into account. To address the problem of tuning the large sets of parameters of such systems, learning to rank methods were developed. These methods use machine learning techniques to tune the parameters of an IR system automatically. Learning to rank for IR is an active research area, and many approaches have been proposed and refined in recent years (Liu, 2009). In this section we give a brief overview of the types of learning to rank methods developed for IR settings.

For the purpose of this thesis, we assume that learning to rank for IR approaches require a feature-based representation, where feature vectors encode characteristics of a query, a document, and the relationship between the query and the document. Such feature-based representations enable generalization across documents and queries. The goal of the learner is then to find generalizable patterns in how to combine ranking features to improve search results (e.g., according to some IR evaluation metric). This leads to the following problem formulation for supervised learning to rank. The input is provided as samples of the form $(\mathbf{x}, y, q)$. Here, $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is an n-dimensional feature vector that represents the relationship between a document and a query; y denotes the ground truth relevance label of the document for a given query; finally, $q \in \mathbb{Z}$ indicates to which query the sample belongs. The ground truth label y can be obtained from a trained annotator, but it can also be inferred from user behavior (§2.3).

The vast majority of learning to rank for IR approaches are developed for the *supervised* setting. In this setting, training data in the form of a representative sample of queries, documents, and relevance judgments is assumed. The specific form and semantics of the input (feature vectors) and output (predicted labels) differ between supervised learning to rank approaches. Following (Cao et al., 2007), three broad types are distinguished, namely *pointwise*, *pairwise*, and *listwise* approaches (Cao et al., 2007; Liu, 2009). Distinguishing between these types is helpful, as it provides insights into the characteristics of learning approaches, such as the type of *loss function* or optimization goal that can be formulated. In turn, this allows conclusions about their effectiveness and efficiency.

Pointwise learning to rank takes as input the feature vectors $\mathbf{x}$ for individual documents (Liu, 2009), and learns a mapping to the ground truth labels y. Depending on the domain of y, standard supervised machine learning approaches can be used. For example, binary relevance scores (i.e., predicting whether a document is relevant or not) can be learned using standard classification approaches (Nallapati, 2004), and regression approaches can be used to learn continuous relevance scores (i.e., the degree of relevance of a document) (Cossock and Zhang, 2006; Fuhr, 1989). The loss function depends on the specific approach chosen, but could be the zero-one loss in the case of classification, or the squared error in the case of regression. A disadvantage of both formulations is that they do not correspond well to the IR ranking problem. In learning to rank for IR, the order in which documents are placed is crucial, while an exact prediction of relevance values is not. It is possible to show that perfect ranking performance can be achieved even when classification or regression-type losses are high. In addition, IR training sets are often highly imbalanced (there can be orders of magnitude more non-relevant documents than relevant documents), making learning difficult. Advantages of pointwise approaches are their low complexity when compared to pairwise and listwise approaches, and that existing classification or regression approaches can be applied directly. Extensions of the pointwise approach include ordinal regression, where a mapping to output scores and thresholds to distinguish separate relevance levels are learned simultaneously. For example, PRank is a popular and effective approach in settings where predictions of absolute relevance labels are required (Crammer et al., 2001).

Pairwise learning to rank approaches operate on pairs of documents, i.e., they take as input pairs of document vectors for a given query $(\mathbf{x}_{1,q}, \mathbf{x}_{2,q}) \in \mathbb{R}^n \times \mathbb{R}^n$ (Liu, 2009). These pairs are mapped to binary labels, e.g., $y \in \{-1, +1\}$, that indicate whether the two documents are presented in the correct order, or should be switched. This problem can be reduced to binary classification by transforming the input to a single combined feature vector $\mathbf{x} = \mathbf{x}_{1,q} - \mathbf{x}_{2,q}$. In this case, the loss function could be based on the classification errors on all document pairs. Optimizing for this loss function may again result in a mismatch with ranking performance, because in IR evaluation metrics are much more sensitive to ranking changes at the top of a result list, than to changes at the bottom of a result list. In contrast, simply counting classification errors would consider all switches of relevant and non-relevant documents equally important. Also, queries with many associated candidate documents may skew results, as IR metrics are typically averaged with equal weights per query. The complexity of the pairwise approach is higher than for the pointwise approach (quadratic in the number of documents if all possible document pairs are considered), but sampling approaches have been shown to be highly effective and efficient (Sculley, 2009). Finally, depending on the form of the learned function, deriving a final ranking from predictions of pairwise preferences may be hard. However, a major advantage of the pairwise approach over the pointwise approach is that it abstracts from specific relevance scores and instead focuses on the relative order of (pairs of) documents. A widely known and effective approach to pairwise learning to rank is RankSVM, a support vector machine approach to minimizing the pairwise hinge loss (Herbrich et al., 1999; Joachims, 2002). Related approaches are developed in the area of preference learning (Fürnkranz and Hüllermeier, 2010).

Listwise learning to rank operates on complete result rankings (Liu, 2009). These approaches take as input the n-dimensional feature vectors of all m candidate documents for a given query $(\mathbf{x}_{1,q}, \ldots, \mathbf{x}_{m,q}) \in \mathbb{R}^{n \times m}$, and learn to predict either the scores for all candidate documents, or complete permutations of documents. The loss function for such an approach can be an IR evaluation measure, although these can be hard to optimize directly, as they are non-smooth and non-differentiable. Alternatives include smooth approximations of such measures (e.g., SoftRank (Taylor et al., 2008)), or, when ground truth is provided in the form of ranked lists, measures of the difference between predicted rankings and the ground truth (e.g., ListNet (Cao et al., 2007)). Listwise approaches have the advantage that they can directly optimize for high ranking performance, but their complexity can be high. Listwise learning to rank approaches are considered the state-of-the-art, as evidenced by the winning approach of the Yahoo! Learning to Rank Challenge[3] (Burges et al., 2011). Best performance was achieved by an ensemble that combines listwise models, including LambdaRank (Burges et al., 2005; Donmez et al., 2009) and LambdaMART (Burges et al., 2006), which optimize listwise measures directly using gradient descent and boosted decision trees.

As in other supervised learning settings, supervised learning to rank for IR methods typically assume that a representative set of training data (including judgments) is available at training time, so that characteristics of the data can be estimated from this set. This labeled data is most often obtained through manual relevance judgment, a process that is often expensive and, because relevance judges may interpret queries differently from actual users, may not accurately capture users' preferences (Sanderson, 2010) (cf., situational relevance, §2.1). A number of *semi-supervised learning* methods have been proposed more recently, which can, in addition to expensive labeled data, take into account unlabeled sample data, for example as a means of regularization (Szummer and Yilmaz, 2011; Tsivtsivadze et al., 2012).

Both supervised and semi-supervised learning to rank approaches work offline. They use provided training data to learn a ranking function that is expected to generalize well to new data drawn from the same distribution as the training data. Once deployed, they do not continue to learn. In contrast, online methods hold the promise of allowing learning while interacting with users of the retrieval system.

The most common approach for learning without prior labels is *active learning*, where the learner is initially provided with an unlabeled training sample, and can request labels for selected samples from an annotator or relevance judge. The focus of these methods is to reduce manual labeling effort; however, they are not designed to learn from natural user interactions. Active learning approaches have been developed to request labels for queries and documents so that they gain as much information as possible from each labeled instance. Xu et al. (2007) present an algorithm that learns a linear combination of features based on relevance, document density, and diversity, which is then used to select documents for which to obtain feedback. Similarly, Xu and Akella (2008) follow a probabilistic approach that selects documents expected to minimize model variance. Donmez and Carbonell (2009) apply active learning to two state-of-the-art learning to rank algorithms, RankSVM and RankBoost. Their approach selects the training instances expected to have the largest effect on the current model.

[3]See `http://learningtorankchallenge.yahoo.com/` for details.

An interesting extension of the active learning paradigm is a recently proposed *co-active* learning algorithm (Shivaswamy and Joachims, 2012). It assumes that both system and user actively explore possible solutions to speed up learning. Interactions are modeled such that the system presents an initial ranked list, which is then improved by the user. It was shown that feedback provided in this way can lead to effective learning.

In contrast to offline (semi-)supervised learning, we address learning to rank in an online setting, where a system learns directly from interactions with the user. In this setting, labeled training data is not provided but needs to be collected through interaction with the user (Yue and Joachims, 2009). In contrast to active learning, feedback is not explicitly requested, but has to be inferred from natural user interactions. No training data is required before deploying the system (but any existing data could be used for bootstrapping the system), and the system is expected to transparently adapt to its users' true preferences. Online learning to rank for IR naturally maps to problem formulations developed in the area of reinforcement learning (§2.4).

The main challenges that need to be addressed by online learning to rank for IR approaches include the quality of available feedback (e.g., when inferring feedback from click data, see §2.3, addressed in Chapters 4 and 5), and the need to learn quickly and reliably from the available feedback while maintaining high result quality while learning (addressed in Chapters 6 and 7).

Our work builds on existing pairwise and listwise online learning to rank for IR approaches as follows. A first evaluation of RankSVM in an online setting demonstrated that learning from implicit feedback is possible in principle (Joachims, 2002). How to best collect feedback for effective learning from implicit feedback has so far not been examined further, but we hypothesize that online approaches need to explore to learn effectively. Our work on pairwise online learning to rank is based on the approach in (Joachims, 2002), which is detailed below (Algorithm 4 in §2.5.1).

Two of the methods for online learning to rank for IR that have been proposed so far perform listwise learning, meaning that they learn from probabilistic comparisons between pairs of candidate rankers using listwise feedback (Yue and Joachims, 2009; Yue et al., 2012). A first such method, Dueling Bandit Gradient Descent (DBGD) was proposed by (Yue and Joachims, 2009). This method implements stochastic gradient descent over a large or infinite space of ranking solutions. Alternatively, algorithms based on multi-armed bandit formulations have been developed to efficiently find the best ranking solutions of a given set (Yue et al., 2012). Our work on listwise online learning to rank is based on the DBGD algorithm (Algorithm 5 in §2.5.2).

2.3 Click Data and other Types of Implicit Feedback

In the previous section we gave an overview of online learning to rank approaches, designed to learn from user behavior. A crucial component of such an approach is its feedback mechanism, i.e., how to interpret user behavior to provide useful information for learning. In this section we give a brief overview of approaches for leveraging user behavior to improve retrieval.

The earliest method for integrating user feedback with retrieval approaches is the relevance feedback approach introduced by Rocchio (1971). This approach allows users to indicate which returned documents are relevant and/or non-relevant to their information need, and extracts information from the labeled documents to devise a more specific query. Relevance feedback approaches have continued to evolve throughout the past decades. A thorough review is provided in (Ruthven and Lalmas, 2003).

Relevance feedback is an example of *explicit feedback*. Explicit feedback is not part of users' natural interactions towards achieving their (search) goal, but rather it is provided with the sole purpose of improving retrieval performance. It is similar to relevance judgments made by professional judges in that it is reliable (as it is consciously provided to improve system performance) but expensive (it requires users' time and effort).

In contrast to explicit feedback, *implicit feedback* is inferred from users' natural interactions with a (retrieval) system. Approaches for improving retrieval performance using implicit feedback are based on the idea that user interactions can provide some information about user satisfaction, e.g., with the relevance of presented search results. Implicit feedback can include all aspects of recorded user behavior, such as clicks, mouse movement, dwell time, etc. Compared to explicit feedback, obtaining implicit feedback is much cheaper, as it is a by-product of natural user interactions, and requires no additional time or cognitive effort of the user. On the other hand, implicit feedback is typically much noisier than explicit feedback, and therefore its interpretation and use are much more difficult. Surveys and further references on the use of implicit feedback in retrieval are provided in (Fox et al., 2005; Kelly and Teevan, 2003).

In this thesis, we focus on online learning to rank using *click-through data*. Click data is a side-product of natural user interactions. It is abundant in frequently-used search applications, and (to some degree) reflects user behavior and preferences. Clicks are part of the natural interaction between users and (web and other) search engines, and in comparison to other types of feedback can be collected in large quantities at very low cost. Consequently, a large body of work has focused on using click behavior to infer information about users' satisfaction with the search results (Carterette and Jones, 2008; Chapelle and Zhang, 2009; Dupret et al., 2007; Kamps et al., 2009; Radlinski et al., 2008b; Wang et al., 2009) and to improve search result quality (Agichtein et al., 2006; Boyan et al., 1996; Dou et al., 2008; Ji et al., 2009; Joachims, 2002; Jung et al., 2007; Shen et al., 2005).

As for all forms of implicit feedback, a challenge when using click data is how to accurately interpret it. For instance, top-ranked web search results are clicked much more frequently than lower-ranked results, even in the absence of a strong difference in relevance (Joachims et al., 2007). Jung et al. (2007) found that click data does contain useful information, but that variance is high. They propose aggregating clicks over search sessions and show that focusing on clicks towards the end of sessions can improve relevance predictions. Similarly, Scholer et al. (2008) found that click behavior varies substantially across users and topics, and that click data is too noisy to serve as a reliable measure of absolute relevance. Fox et al. (2005) found that combining several implicit indicators can improve accuracy, though it remains well below that of explicit feedback. In particular, evaluation methods that interpret clicks as absolute relevance judgments in more broadly used settings such as literature search, web search, or search on Wikipedia, were found to be rather unreliable, due to large differences in click behavior between users and search topics (Kamps et al., 2009; Radlinski et al., 2008b). Finally, click behavior was found to be affected by visual aspects of result presentation (§2.3.2).

Nonetheless, in some applications, click data has proven reliable. In searches of expert users who are familiar with the search system and document collection, clicks can be as reliable as purchase decisions (Hofmann et al., 2010b; Zhang and Kamps, 2010). Methods for optimizing the click-through rates in ad placement (Langford et al., 2008) and web search (Radlinski et al., 2008a) have also learned effectively from click data.

Methods that use implicit feedback to infer the relevance of specific document-query pairs have also proven effective. Shen et al. (2005) show that integrating click-through information for query-document pairs into a content-based retrieval system can improve retrieval performance substantially. Agichtein et al. (2006) demonstrate dramatic performance improvements by re-ranking search results based on a combination of implicit feedback sources, including click-based and link-based features.

The quickly growing area of click modeling develops and investigates models of users' click behavior (Chapelle and Zhang, 2009; Dupret and Liao, 2010; Dupret et al., 2007). These models are trained *per query* to predict clicks and/or relevance of documents that have not been presented to users at a particular rank, or that have not been presented at all for the given query. An advantage of click models is that they directly model absolute relevance grades of individual documents. However, it is not yet clear to what degree they can complement or replace editorial judgments for evaluation. Extensions of click models combine inferred relevance with editorial judgments. These extensions have been found to effectively leverage click data to allow more accurate evaluations with relatively few explicit judgments (Carterette and Jones, 2008; Ozertem et al., 2011). Recently developed evaluation metrics that incorporate insights gained from click models (Chapelle et al., 2009; Moffat and Zobel, 2008) provide new possibilities for combining click data and editorial judgments, further bridging the gap between click-based and traditional retrieval evaluation. The click models mentioned above can be reused to some degree but, unlike our methods, do not generalize across queries.

Since clicks and other implicit feedback vary so much across queries, it is difficult to use them to learn models that generalize across queries. To address this problem, Joachims (2002) proposes to interpret clicks not as *absolute* feedback (e.g., whether or not a document is relevant), but *relative* to its context (whether a clicked document is more or less relevant than a preceding non-clicked document). This interpretation was successfully demonstrated by Joachims (2002), and forms the basis of our research on document-pairwise online learning to rank (Chapter 6, esp., §6.1.1).

A particularly promising approach to interpreting click-through data are *interleaved comparison methods* (Radlinski et al., 2008b). These methods use clicks on interleaved result lists to infer relative feedback on ranking functions, and have been shown to provide reliable comparisons in large-scale web search evaluations (Chapelle et al., 2012; Radlinski and Craswell, 2010). A more detailed discussion of interleaved comparison methods and an overview of existing methods are provided in the next section. The research presented in this thesis builds on these methods.

2.3.1 Interleaved Comparison Methods

Interleaved comparison methods use click data to compare ranking functions. These methods are quickly gaining popularity as a form of online evaluation, a complement to TREC-style evaluations. Compared to TREC-style evaluations, which require expensive manual relevance judgments, interleaved comparison methods rely only on click data, which can be collected cheaply and unobtrusively. Furthermore, since this data is based on the behavior of real users, it more accurately reflects how well their actual information needs are met (Radlinski et al., 2008b). Previous work demonstrated that two rankers can be successfully compared using click data in practice (Chapelle et al., 2012).

From the viewpoint of learning to rank, interleaved comparison methods are interesting, as they infer listwise feedback from clicks, which can enable online listwise learning to rank approaches. Methods for learning to rank from such feedback have been proposed (Yue and Joachims, 2009; Yue et al., 2012), but our work is the first to empirically confirm that online learning to rank is possible using the relative, listwise feedback obtained from interleaved comparison methods (Chapter 6).

At a high level, interleaved comparison methods compare rankers in two steps, one interleaving step and one comparison step. During the interleaving step, ranked result lists for a given query are obtained from the two competing rankers. From these, an interleaved result list is generated in such a way that position bias between the two rankers is minimized (Radlinski et al., 2008b). The interleaved result list is presented to the user and clicks are recorded. Then, during the comparison step, the observed clicks are associated with the original rankers to infer which ranker the user would prefer.

Interleaving involves showing each user results returned by both retrieval functions. This allows the user's selection process to provide evidence as to which retrieval function, in expectation, returns relevant results more often. By doing this direct comparison, interleaving has been shown to be more sensitive than alternative approaches (Radlinski et al., 2008b). Radlinski and Craswell (2010) compare the reliability and sensitivity of TD to judgement-based evaluation in a web search setting, and Chapelle et al. (2012) provide a detailed comparison and evaluation of several interleaving approaches. Alternative interleaving approaches have also been proposed (Joachims, 2003) (cf., BI, below), as well as alternative scoring approaches (He et al., 2009) (cf. DC, below). In their most recent work (contemporary with this thesis) Radlinski and Craswell (2013) formulate interleaving as an optimization problem, and explore several rank-based scoring functions with the goal of increasing the sensitivity of interleaved comparisons.

Below, we introduce the three existing interleaved comparison methods, BI, TD, and DC. All three methods are designed to compare pairs of rankers $(\mathbf{l}_1(q),\ \mathbf{l}_2(q))$.[4] Here, rankers are deterministic functions that, given a query q, produce a ranked list of documents d. Given $\mathbf{l}_1$ and $\mathbf{l}_2$, interleaved comparison methods produce outcomes $o \in \{-1, 0, 1\}$ that indicate whether the quality of $\mathbf{l}_1$ is judged to be lower, equal to, or higher than that of $\mathbf{l}_2$, respectively. For reliable comparisons, these methods are typically applied over a large number of queries and the individual outcomes are aggregated. However, in this section, we focus on how interleaved comparison methods compute individual outcomes. We analyze these interleaved comparison methods in §4.1, and propose a new, probabilistic interleaved comparison method in §4.2. We present learning approaches based on interleaved comparisons, and empirical evaluations in Chapters 6 and 7.

[4]If it is clear from the context which q is referred to, we simplify our notation to $\mathbf{l}_1$ and $\mathbf{l}_2$.

Balanced Interleave

BI (Joachims, 2003; Radlinski et al., 2008b) generates an interleaved result list **l** as follows (cf., Algorithm 1, lines 3–12). First, one of the result lists is randomly selected as the starting list and its first document is placed at the top of **l**. Then, the non-starting list contributes its highest-ranked document that is not already part of the list. These steps repeat until all documents have been added to **l**, or until it has the desired length. Next, the constructed interleaved list **l** is displayed to the user, and the user's clicks on result documents are recorded. The clicks **c** that are observed are then attributed to each list as follows (lines 13–17). For each original list, the rank of the lowest-ranked document that received a click is determined, and the minimum of these values is denoted as v. Then, the clicked documents ranked at or above v are counted for each original list. The list with more clicks in its top v is deemed superior. The lists tie if they obtain the same number of clicks.

Algorithm 1 Interleaved comparison with BI, following (Chapelle et al., 2012).

```
 1: Input: $\mathbf{l}_1, \mathbf{l}_2$
 2: $\mathbf{l} = []; i_1 = 0; i_2 = 0$
 3: $first\_1 = random\_bit()$
 4: while $(i_1 < len(\mathbf{l}_1)) \wedge (i_2 < len(\mathbf{l}_2))$ do
 5:    if $(i_1 < i_2) \vee ((i_1 == i_2) \wedge (first\_1 == 1))$ then
 6:       if $\mathbf{l}_1[i_1] \notin \mathbf{l}$ then
 7:          $append(\mathbf{l}, \mathbf{l}_1[i_1])$
 8:       $i_1 = i_1 + 1$
 9:    else
10:       if $\mathbf{l}_2[i_2] \notin \mathbf{l}$ then
11:          $append(\mathbf{l}, \mathbf{l}_2[i_2])$
12:       $i_2 = i_2 + 1$
    // present $\mathbf{l}$ to user and observe clicks $\mathbf{c}$, then infer outcome (if at least one click was observed)
13: $d_{max}$ = lowest-ranked clicked document in $\mathbf{l}$
14: $v = min\{j : (d_{max} = \mathbf{l}_1[j]) \vee (d_{max} = \mathbf{l}_2[j])\}$
15: $c_1 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_1[1..v]\}$
16: $c_2 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_2[1..v]\}$
17: return $-1$ if $c_1 > c_2$ else $1$ if $c_1 < c_2$ else $0$
```

Team Draft

The alternative interleaved comparison method TD (Radlinski et al., 2008b) creates an interleaved list following the model of "team captains" selecting their team from a set of players (cf., Algorithm 2). For each pair of documents to be placed on the interleaved list, a coin flip determines which list gets to select a document first (line 4). It then contributes its highest-ranked document that is not yet part of the interleaved list. The method also records which list contributed which document in an *assignment* $\mathbf{a}$ (lines 7, 11). To compare the lists, only clicks on documents that were contributed by each list (as recorded in the assignment) are counted towards that list (lines 12–14), which ensures

Algorithm 2 Interleaved comparison with TD, following (Chapelle et al., 2012).

```
1: **Input**: $\mathbf{l}_1, \mathbf{l}_2$
2: $\mathbf{l} = []; \mathbf{a} = []$
3: **while** $(\exists i : \mathbf{l}_1[i] \notin \mathbf{l}) \vee (\exists i : \mathbf{l}_2[i] \notin \mathbf{l})$ **do**
4:     **if** $count(\mathbf{a}, 1) < count(\mathbf{a}, 2) \vee (rand\_bit() == 1)$ **then**
5:         $j = min\,\{i : \mathbf{l}_1[i] \notin \mathbf{l}\}$
6:         $append(\mathbf{l}, \mathbf{l}_1[j])$
7:         $append(\mathbf{a}, 1)$
8:     **else**
9:         $j = min\,\{i : \mathbf{l}_2[i] \notin \mathbf{l}\}$
10:        $append(\mathbf{l}, \mathbf{l}_2[j])$
11:        $append(\mathbf{a}, 2)$
   // present $\mathbf{l}$ to user and observe clicks $\mathbf{c}$, then infer outcome
12: $c_1 = len\,\{i : c[i] = true \wedge \mathbf{a}[i] == 1\}$
13: $c_2 = len\,\{i : c[i] = true \wedge \mathbf{a}[i] == 2\}$
14: **return** $-1$ **if** $c_1 > c_2$ **else** $1$ **if** $c_1 < c_2$ **else** $0$
```

that each list has an equal chance of being assigned clicks. Again, the list that obtains more clicks wins the comparison. Recent work demonstrates that TD can reliably identify the better of two rankers in practice (Chapelle et al., 2012; Radlinski and Craswell, 2010).

Document Constraints

While BI and TD directly aggregate clicks to detect preferences between rankers, He et al. (2009) hypothesize that the efficiency of interleaved comparison methods can be improved if methods also take into account the preference relations between documents that can be inferred from clicks. Based on this hypothesis, the authors propose an approach that we refer to as DC (cf., Algorithm 3).

Result lists are interleaved and clicks observed as for BI (lines 3–12). Then, following (Joachims, 2002), the method infers constraints on pairs of individual documents, based on their clicks and ranks. Two types of constraints are defined: (1) for each pair of a clicked document and a higher-ranked non-clicked document, a constraint is inferred that requires the former to be ranked higher than the latter; (2) a clicked document is inferred to be preferred over the next unclicked document.[5] The method compares the inferred constraints to the original result lists and counts how many constraints are violated by each. The list that violates fewer constraints is deemed superior. Though more computationally expensive, this method proved more reliable than either BI or TD on synthetic data (He et al., 2009).

[5]Variants of this method can be derived by using only the constraints of type (1), or by using an alternative constraint (2) where only unclicked documents are considered that are ranked immediately below the clicked document. In preliminary experiments, we evaluated all three variants and found the one using constraints (1) and (2) as stated above to be the most reliable. Note that only constraints of type (1) were used in earlier work (Hofmann et al., 2011c, 2012b).

Algorithm 3 Interleaved comparison with DC, following (He et al., 2009).

```
1: **Input**: $\mathbf{l}_1, \mathbf{l}_2$
2: $\mathbf{l} = []; i_1 = 0; i_2 = 0$
3: $first\_1 = random\_bit()$
4: while $(i_1 < len(\mathbf{l}_1)) \wedge (i_2 < len(\mathbf{l}_2))$ do
5:    if $(i_1 < i_2) \vee ((i_1 == i_2) \wedge (first\_1 == 1))$ then
6:       if $\mathbf{l}_1[i_1] \notin \mathbf{l}$ then
7:          $append(\mathbf{l}, \mathbf{l}_1[i_1])$
8:       $i_1 = i_1 + 1$
9:    else
10:      if $\mathbf{l}_2[i_2] \notin \mathbf{l}$ then
11:         $append(\mathbf{l}, \mathbf{l}_2[i_2])$
12:      $i_2 = i_2 + 1$
   // present $\mathbf{l}$ to user and observe clicks $\mathbf{c}$, then infer outcome
13: $v_1 = violated(\mathbf{l}, \mathbf{c}, \mathbf{l}_1)$   // count constraints inferred from $\mathbf{l}$ and $\mathbf{c}$ that are violated by $\mathbf{l}_1$
14: $v_2 = violated(\mathbf{l}, \mathbf{c}, \mathbf{l}_2)$   // count constraints inferred from $\mathbf{l}$ and $\mathbf{c}$ that are violated by $\mathbf{l}_2$
15: return $-1$ if $v_1 < v_2$ else $1$ if $v_1 > v_2$ else $0$
```

2.3.2 Click Bias

While clicks are becoming more popular as a source of preference indications on search results, a number of studies have found that click behavior is affected by bias. In this section we give a brief overview of the types of bias previously found to affect users' click behavior in web search.

In the context of interleaved comparisons, position bias has been addressed. Position bias results from the layout of a search result page. Because users generally expect more relevant items to be listed at the top of a page, and because people are used to reading pages from top to bottom, top-ranked results are typically the most likely to be examined. This phenomenon was first confirmed in eye-tracking studies (Granka et al., 2004; Guan and Cutrell, 2007). Craswell et al. (2008) developed models of user behavior to describe position bias, and described a cascade model to explain this effect. The model was refined in several follow-up studies, e.g., to account for multiple clicks on the same result page (Guo et al., 2009b), and to account for differences in click behavior for different types of queries and search goals (Guo et al., 2009a).

In addition to position bias, which reflects *where* on the page a result was displayed, click behavior is also affected by caption bias, caused by *how* the result was displayed. Clarke et al. (2007) studied caption bias by comparing features and click behavior on pairs of search results. They found that results were more often clicked on when they had longer snippets, shorter URLs, more query terms matching the caption, matches of the whole query as a phrase, if the caption was more readable, or if it contained the term "official" or terms related to images. Yue et al. (2010b) compared click data on result documents that were sampled to minimize position bias using the Fair Pairs approach (Radlinski and Joachims, 2006) to editorial judgments. They identified a bias towards captions that included more highlighted (bold) terms, i.e., items with more bold terms would be clicked more frequently than similar results with fewer bold terms.

Other factors affecting click behavior include the domain of the search result (Ieong et al., 2012), whether or not search results are grouped (e.g., by intent) (Dumais and Cutrell, 2001), page loading time (Wang et al., 2009), the amount of context shown in the snippet (Tombros and Sanderson, 1998), and the relation between task and result caption (Cutrell and Guan, 2007).

In this thesis, we address the effects of visual factors on click behavior, in particular in a web search setting, in Chapter 5. In §5.2.4, we report on the effects of such caption bias on interleaving experiments. While our model focuses on caption bias, our approach is generally applicable to other sources of click bias. In the remainder of this thesis, we compensate for position bias using interleaved comparisons (Chapters 4 and 7) or by balancing exploration and exploitation (Chapter 6), and assume that other sources of click bias have been compensated for.

2.4 Reinforcement Learning

In this thesis, we formulate online learning to rank as an RL problem, a problem in which an *agent* learns from interactions with an *environment* (Kaelbling et al., 1996; Sutton and Barto, 1998) (cf., Chapter 3). Using this formulation allows us to draw from the ideas and solutions developed in RL. First, we give an overview of the terminology and concepts from RL that are important for the IR problem formulation proposed in Chapter 3. Then we outline standard RL concepts and solutions that form the basis for methods developed in this thesis, including strategies for balancing exploration and exploitation (§2.4.2), and off-policy evaluation (§2.4.3).

In RL, an agent interacts with an unknown environment over a series of timesteps, observing the *state* of the environment, taking *actions*, and receiving *rewards*, which can be positive, negative, or zero (Kaelbling et al., 1996). For example, a robot navigating an unfamiliar maze can take actions to move in different directions and might receive positive reward for reaching a goal and negative reward for using a scarce resource such as battery power. The distinguishing characteristic of RL problems is that the agent learns through trial and error (Sutton and Barto, 1998). In this setting, the agent can only observe the rewards for the actions it selected, meaning that it is never shown any examples of the optimal action for any situation, as is the case in e.g., supervised learning.

The goal of the agent in an RL problem is to maximize *cumulative reward*, accumulated while interacting with the environment. How cumulative reward is defined depends on whether the task is formulated as a *finite* or *infinite horizon* problem. In finite horizon problems, the interaction between the agent and its environment is limited to a fixed number of timesteps T. For example, the task of playing soccer could be modeled as a finite horizon problem that terminates when time expires. In this case, the cumulative reward C is simply the sum of rewards received until termination:

$$C = \sum_{i=1}^{T} r_i,$$

where r_i is the reward received on the ith timestep.

In infinite horizon problems, the interaction between the agent and its environment continues indefinitely. E.g., the task of managing resources in a factory can be modeled

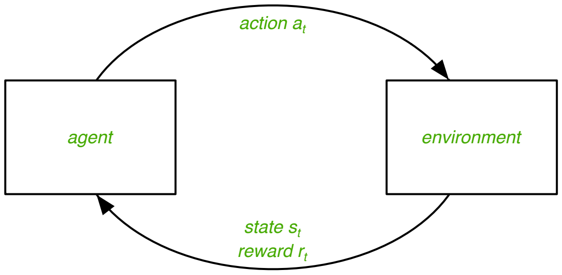


Figure 2.1: Generic RL problem and terminology. In *contextual bandit problems* states are independent of previous actions.

as an infinite horizon problem, since factories often remain open indefinitely. One issue with infinite horizon problems is the *infinitely delayed splurge*: since there are always infinitely many timesteps to go, the agent always explores, confident that enough time remains to exploit. To address the issue, infinite horizon problems typically include a *discount factor* $\gamma \in [0, 1)$ which weights immediate rewards higher than future rewards. Hence, the agent has an incentive to balance exploration and exploitation, instead of always exploring. Here, cumulative reward is defined as the discounted infinite sum of rewards:

$$C = \sum_{i=1}^{\infty} \gamma^{i-1} r_i. \tag{2.2}$$

When $\gamma = 0$, the agent cares only about maximizing immediate rewards through exploitation. As γ approaches 1, future rewards take on greater importance and the agent's incentive to explore increases. One way to interpret the discount factor is to suppose that there is a $1 - \gamma$ probability that the task will terminate at each timestep. Rewards are thus weighted according to the probability that the task will last long enough for them to occur. This is the formulation used in this thesis (cf., §3.2).

An agent's behavior is determined by its *policy*, which specifies what action it should take in each state. Solutions to finding an optimal policy fall in two categories. First, *policy-search* methods use optimization techniques such as gradient methods (Sutton et al., 2000) or evolutionary computation (Moriarty et al., 1999) to directly search the space of policies for those accruing maximal reward. Second, *value-function* methods work by estimating the expected long-term reward for taking an action in a state and behaving optimally thereafter (Sutton, 1988). Given an optimal value function, an optimal policy can be easily derived by selecting in each state the *greedy action*: the one that maximizes this value function. The methods explored in this thesis are based on policy search.

2.4.1 Contextual Bandit Problems

Particularly relevant to this thesis are methods for tackling *contextual bandit problems* (also known as *bandits with side information* or *associative RL* (Auer, 2003; Barto et al., 1981; Langford and Zhang, 2008)), a well-studied type of RL problem (Auer, 2003;

Barto et al., 1981; Langford and Zhang, 2008; Strehl et al., 2006), as they have been successfully applied to problems similar to learning to rank for IR (Agarwal et al., 2008; Langford et al., 2008; Li et al., 2010, 2011; Radlinski et al., 2008a; Zhang et al., 2003).

A contextual bandit problem is a special case of an RL problem in which states are independent of the agent's actions. In other words, the agent has no control over the states to which it transitions (Figure 2.1). Instead, its actions affect only its immediate reward. A difference between typical contextual bandit formulations and online learning to rank for IR is that in IR (absolute) reward cannot be observed directly. Instead, feedback for learning can be inferred from observed user interactions as noisy relative preference indications (cf., §2.3).

Contextual bandit formulations have proved successful in applications that are similar to online learning to rank for IR, in cases where implicit feedback can be interpreted in absolute terms (e.g., in cases where maximizing the click-through rate can be assumed to lead to good task performance). One solution is to reduce the contextual bandit problem to several multi-armed bandit problems (a multi-armed bandit problem has only one state or context), so that a different solution is learned for each context. For example, Langford et al. (2008) consider the ad placement application. Given a website, their algorithm learns the value of placing each of a set of candidate ads on the website. Similarly, Radlinski et al. (2008a) consider how to learn diverse document lists such that different information needs are satisfied; they present an algorithm for doing so that balances exploration and exploitation. Our solutions differ from this type of approach in that context is taken into account by using a feature-based representation, which allows them to generalize over queries (cf. §2.2).

Another widely-studied application area of related approaches is news recommendation, where news stories are selected for a user population or for individual users. Work in this area has focused on learning approaches (Agarwal et al., 2008; Li et al., 2010), and methods for offline evaluation (Li et al., 2011). Finally, an application to adaptive filtering is presented by Zhang et al. (2003). However, like other RL algorithms, these methods all assume access to absolute feedback. For example, in ad placement, clicks can be interpreted as absolute feedback because they are directly correlated with the value of the ad-website pair (assuming a pay-per-click model). Since interpreting clicks as absolute feedback is problematic in online IR settings (Joachims et al., 2007; Radlinski et al., 2008b) (§2.3), these methods are not directly applicable. While in related areas implicit feedback can often be interpreted as absolute reward, this is not possible in our setting.

2.4.2 Balancing Exploration and Exploitation

A central challenge of RL is the problem of balancing exploration and exploitation.[6] As the agent's environment is initially unknown, and the agent only receives feedback (reward) for the actions it tries, the agent needs to try out new actions to learn about their effects. In addition, it is not enough for the agent to discover a good solution by

[6]In the related area of *search and optimization*, exploration and exploitation are used in a different sense. There, exploration means global search (over a large part of a solution space) and exploitation means local search (close to previously identified optima) (Chen et al., 2009). In this thesis we always refer to exploration and exploitation in the RL meaning, as described in this section.

the end of learning (as is the case in supervised learning). Rather, it should maximize the cumulative reward it receives during its interaction with the environment. For this reason, balancing exploration and exploitation is crucial. The agent needs to try out new solutions to be able to learn from the observed feedback, and it needs to exploit what it has already learned to ensure high reward.[7]

Most approaches for balancing exploration and exploitation in RL have been developed for value-function methods. For these methods, it is possible to compute a *Bayes-optimal* exploration strategy (Poupart et al., 2006; Strens, 2000), but doing so is typically intractable. Many approaches for heuristically balancing exploration and exploitation exist (Kaelbling, 1993; Sutton and Barto, 1998). E.g., in *ϵ-greedy* exploration (Watkins, 1989), the agent selects an action with probability ϵ at each step. With probability $1 - \epsilon$, it selects the *greedy* action, i.e., the one with highest currently estimated value.

Policy-search methods are by nature exploratory, so maximizing cumulative performance requires supplementing them with mechanisms for properly balancing exploration and exploitation. To this end, exploration heuristics developed for value-function methods have been successfully adapted for policy search (Whiteson and Stone, 2006b).

Because balancing exploration and exploitation is considered important for optimizing performance while learning in an RL setting, we hypothesize that similar benefits can be achieved in online learning to rank for IR. We investigate this hypothesis in Chapter 6.

2.4.3 Off-policy Evaluation

One part of this thesis explores the idea of reusing previously collected data for interleaved comparisons (Chapter 4) and online learning to rank for IR (Chapter 7). Such data reuse was not possible with previous interleaved comparison methods. However, from an RL perspective, our work is related to *off-policy learning* (Precup et al., 2000; Sutton and Barto, 1998). Off-policy learning was developed in the RL community to address settings where interactions with the environment (to evaluate a new policy) is expensive (e.g., due to cost of material, such as a robot, or because repeating many real-time interactions can take a long time). When data from earlier policy evaluations is available, off-policy methods estimate the value of new policies based on this data.

Algorithms for off-policy evaluation have been developed for tasks similar to IR, namely news recommendation (Dudík et al., 2011; Li et al., 2011) and ad placement (Langford et al., 2008; Strehl et al., 2010). In both settings, the goal is to evaluate the policy of an agent (recommendation engine, or ad selector) that is presented with a context (e.g., a user profile, or website for which an ad is sought), and selects from a set of available actions (news stories, ads). Off-policy learning in this context is hard because the data is sparse, i.e., not all possible actions were observed in all possible contexts. Solutions to this problem are based on randomization during data collection (Li et al., 2011), approximations for cases where exploration is non-random (Langford et al., 2008; Strehl et al., 2010), and combining biased and high-variance estimators to obtain more robust

[7]Balancing exploration and exploitation plays an important role in many areas, such as sequential experimental design and in the multi-armed bandit work coming from the applied probability community. Early work includes (Robbins, 1952), with an important breakthrough by Gittins (1979). A recent overview can be found in (Mahajan and Teneketzis, 2008). Exploration and exploitation have also been extensively studied as fundamental principles of human and animal decision-making behavior (Cohen et al., 2007).

results (Dudík et al., 2011).

Though sparse data is also a problem in IR, existing solutions to off-policy evaluation are not directly applicable. These methods assume reward can be directly observed (e.g., in the form of clicks on ads). Since clicks are too noisy to be treated as absolute reward in IR (Kamps et al., 2009; Radlinski et al., 2008b), only relative feedback can be inferred. In §4.2.3, we consider how to reuse historical data for interleaved comparison methods that work with implicit, relative feedback.

However, one tool employed by existing off-policy methods that is applicable to our setting is a statistical technique called *importance sampling* (MacKay, 1998; Precup et al., 2000). Importance sampling can be used to estimate the expected value $E_T[f(X)]$ under a *target distribution* P_T when data was collected under a different *source distribution* P_S. The importance sampling estimator is:

$$E_T[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)\frac{P_T(x_i)}{P_S(x_i)}, \tag{2.3}$$

where f is a function of X, and the x_i are samples of X collected under P_S. These are then reweighted according to the ratio of their probability of occurring under P_T and P_S. This estimator can be proven to be statistically *sound* (i.e., unbiased and consistent, cf., Definition 4.1.3 in §4.1) as long as the source distribution is non-zero at all points at which the target distribution is non-zero (MacKay, 1998).

Importance sampling can be more or less efficient than using the target distribution directly, depending on how well the source distribution focuses on regions important for estimating the target value. In §4.2.3, we use importance sampling to derive an unbiased and consistent estimator of interleaved comparison outcomes using historical data. In Chapter 7, we show that this estimator allows effective reuse of historical interaction data in online learning to rank for IR.

2.5 Baseline Online Learning to Rank Approaches

Below, we detail our two baseline algorithms for online learning to rank for IR, which form the basis of our work on pairwise and listwise online learning to rank for IR in Chapters 6 and 7.

Both approaches are based on a feature-representation of document-query relations, i.e., input consists of the feature vectors of the m candidate documents for a given query $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$.[8] Also, both learn a weight vector $\mathbf{w}$ for linear-weighted combinations of these features. At any point t during learning, a ranked list can be obtained from a current weight vector $\mathbf{w}_t$ for a given query by scoring the candidate documents using $\mathbf{s} = \mathbf{w}_t^T \times (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, and sorting them by their scores. The weight vectors are learned using feedback inferred from user clicks. In the case of the pairwise approach, user behavior is used to infer preferences between document pairs. In the case of the listwise approach, preferences are inferred between complete result lists using interleaved comparison methods (§2.3.1).

[8] In practice, candidate documents are typically collected based on some feature-based criteria, such as a minimum retrieval score.

2.5.1 Learning from Document-Pairwise Feedback

Our first approach builds off a pairwise formulation of learning to rank, and a stochastic gradient descent learner. Document-pairwise approaches model the pairwise relations between documents for a given query. Our formulation of the learning to rank problem from implicit feedback follows Joachims (2002). The learning algorithm is a stochastic gradient descent algorithm, following Zhang (2004) and (Sculley, 2009).

Pairwise preferences are inferred from clicks, following Joachims (2002) (cf., §2.3). For example, assume a query q, in response to which the system returns documents (d_1, d_2, d_3), in this order. If the user clicks on documents d_2 and d_3, but not on d_1, we can infer that $d_2 \succ d_1$ and $d_3 \succ d_1$. From these observations, labeled data could be extracted as $(d_1, d_2, -1)$ and $(d_1, d_3, -1)$.

Given a set of labeled document pairs, we apply the stochastic gradient descent (SGD) algorithm by Zhang (2004, Algorithm 2.1). This algorithm finds an optimal weight vector $\hat{\mathbf{w}}$ that minimizes the empirical loss $L(\mathbf{w}, \mathbf{x}, y)$ given a set $\mathcal{P}$ of labeled training samples, each consisting of a feature vector $\mathbf{x}$ and a label y:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} L(\mathbf{w}, \mathbf{x}_i, y_i) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \right], \tag{2.4}$$

where the last term is a regularization term. Using the hinge loss, i.e., $L(\mathbf{w}, \mathbf{x}, y) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$, the algorithm optimizes the same quantity as RankSVM (Joachims, 2002). It was shown to perform competitively on standard learning to rank data sets in terms of ranking performance with only a fraction of the training time (Sculley, 2009). Here, we follow the implementation provided in sofia-ml,[9] and apply it to pairwise learning by setting $\mathbf{x} = (\mathbf{x}_{1,q} - \mathbf{x}_{2,q})$, where $\mathbf{x}_{1,q}$ and $\mathbf{x}_{2,q}$ are the feature vectors of two candidate documents for a query q.

Combining the above method of inferring pairwise feedback and the pairwise learning method, we obtain our pairwise baseline algorithm (Algorithm 4). It receives as input a document set $\mathcal{D}$, learning rate η, regularizer weight λ, and an initial weight vector $\mathbf{w}_0$. For each observed query q_t, a set of feature vectors $\phi(d_i|q_t)$ is extracted that characterize the relationship between the query and each candidate document $d_i \in \mathcal{D}$. The document feature vectors are then scored using the weight vector learned so far ($\mathbf{w}_{t-1}$), and sorted by this score to obtain an exploitative result list (the best ranking given what has been learned so far).

The constructed exploitative result list is shown to the user, and clicks on any of the result documents are observed. From the observed clicks $\mathbf{c}$, all possible labeled document pairs $\mathcal{P}$ are inferred using the pairwise labeling method described above (Joachims, 2002). The labeled samples in $\mathcal{P}$ are then used to update the weight vector $\mathbf{w}$. For each pair, the loss is obtained by comparing the current solution to the observed label (line 10 in the definition of the hinge loss above). If the labels do not match, or the prediction margin is too small, the weight vector is updated using the update rule $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y(\mathbf{x}_1 - \mathbf{x}_2) - \eta\lambda\mathbf{w}_{t-1}$. Here, we use the unregularized version of this update rule (by setting $\lambda = 0$) and use a small constant η. This formulation was found

[9]Provided online at `http://code.google.com/p/sofia-ml/`.

Algorithm 4 Baseline online learning to rank algorithm for the pairwise setting, based on (Joachims, 2002; Sculley, 2009; Zhang, 2004).

```
1: Input: $\mathcal{D}$, $\eta$, $\lambda$, $\mathbf{w}_0$
2: for query $q_t$ ($t = 1..\infty$) do
3:    $\mathbf{X} = \phi(\mathcal{D}|q_t)$                                  // extract features
      // generate result list
4:    $\mathbf{s} = \mathbf{w}_{t-1}^T\mathbf{X}$
5:    $\mathbf{l} = sort\_descending\_by\_score(\mathcal{D}, \mathbf{s})[1:10]$
6:    Display $\mathbf{l}$ and observe clicked elements $\mathbf{c}$.
7:    Construct all labeled pairs $\mathcal{P} = (\mathbf{x}_1, \mathbf{x}_2, y)$ for $q_t$ from $\mathbf{l}$ and $\mathbf{c}$.
8:    for $(\mathbf{x}_1, \mathbf{x}_2, y)$ in $\mathcal{P}$ do
9:       if $y\mathbf{w}_{t-1}^T(\mathbf{x}_1 - \mathbf{x}_2) < 1.0$ and $y \neq 0.0$ then
10:         $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y(\mathbf{x}_1 - \mathbf{x}_2) - \eta\lambda\mathbf{w}_{t-1}$       // update $\mathbf{w}_t$
```

to show good convergence properties (Zhang, 2004) and resulted in good performance in preliminary experiments.

2.5.2 Dueling Bandit Gradient Descent

Our listwise baseline approach is DBGD, a listwise stochastic gradient descent algorithm proposed in (Yue and Joachims, 2009). It is based on randomized search of the solution space, and uses feedback about the relative quality of result lists. The approach was previously shown to work effectively under smoothness assumptions for this feedback, and was empirically evaluated with stochastic feedback based on true NDCG differences.

DBGD learns weight vectors as shown in Algorithm 5 (Yue and Joachims, 2009). Its first input is a comparison function $f(\mathbf{l}_1, \mathbf{l}_2)$, which compares two result lists $\mathbf{l}_1$ and $\mathbf{l}_2$ using user clicks $\mathbf{c}$ (the return value $o_L \in \mathbb{R}$ indicates whether the quality of the two lists was inferred to be equal ($o_L = 0$), or whether the first ($o_L < 0$) or second ($o_L > 0$) list was inferred to be better; cf., 2.3.1). A second function, $g(\delta, \mathbf{w})$ is provided to generate candidate rankers. The remaining inputs are the step sizes α[10] and δ, and an initial weight vector $\mathbf{w}_0$. An optional parameter θ indicates the maximum amount of most recent historic interaction data that the algorithm should keep in memory for possible reuse. This parameter is set to 0 in the baseline version.

The algorithm learns while interacting with search engine users as follows. At any time, the hypothesized best solution up to that point is maintained as $\mathbf{w}_t$. When a query q_t is observed, a new candidate weight vector $\mathbf{w}'_t$ is generated using $g(\cdot)$ (line 4). Then, result lists for q_t are generated using both the current best ($\mathbf{w}_t$) and the candidate ($\mathbf{w}'_t$) weight vector (line 5; $generate\_list(\cdot)$ generates a result list using a weight vector as shown in lines 4 and 5 of Algorithm 4). The two result lists are compared using $f(\mathbf{l}_1, \mathbf{l}_2)$ (line 6). If $\mathbf{w}'_t$ wins the comparison, $\mathbf{w}_t$ is updated using the update rule $\mathbf{w}_t \leftarrow \mathbf{w}_t + \alpha\mathbf{u}_t$ (line 8). Otherwise, $\mathbf{w}_t$ is not changed. Lines 11–14 of Algorithm 5 shows how historical

[10]Yue and Joachims (2009) use γ to denote the exploitation step size. We use α to avoid confusion with the discount factor γ.

Algorithm 5 Baseline online learning to rank algorithm for the listwise setting, based on (Yue and Joachims, 2009).

```
1: Input: $f(\mathbf{l}_1, \mathbf{l}_2)$, $g(\delta, \mathbf{w})$, $\alpha$, $\delta$, $\mathbf{w}_0$, $\theta$ (default: 0)
2: $\mathbf{h} \leftarrow []$
3: for query $q_t$ ($t \leftarrow 1..\infty$) do
4:    $(\mathbf{w}'_t, \mathbf{u}_t) \leftarrow g(\delta, \mathbf{w}_t)$          // generate candidate ranker
5:    $\mathbf{l}_1 = generate\_list(\mathbf{w_t})$; $\mathbf{l}_2 = generate\_list(\mathbf{w'_t})$
6:    $(o_L, \mathbf{l}, \mathbf{a}, \mathbf{c}) \leftarrow f(\mathbf{l}_1, \mathbf{l}_2)$
7:    if $o_L > 0$ then
8:       $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \mathbf{u}_t$          // update current best ranker
9:    else
10:      $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
      // maintain historical data if needed
11:   if $\theta > 0$ then
12:      if $len(\mathbf{h}) = \theta$ then
13:         $remove(\mathbf{h}, \mathbf{h}[0])$
14:      $append(\mathbf{h}, (\mathbf{l}, \mathbf{a}, \mathbf{c}))$
```

Algorithm 6 $generate\_candidate(\cdot)$ (baseline method for generating candidate rankers, to be used as $g(\delta, \mathbf{w})$ in Algorithm 5).

```
1: Input: $\delta$, $\mathbf{w}$
2: Sample unit vector $\mathbf{u}$ uniformly.
3: $\mathbf{w}' \leftarrow \mathbf{w} + \delta \mathbf{u}$
4: return $(\mathbf{w}', \mathbf{u})$
```

data is recorded if necessary (if $\theta > 0$, cf., Chapter 7 for learning approaches that reuse historical data).

In the baseline version of this algorithm, $generate\_candidate(\cdot)$ is used to generate candidate weight vectors as follows (Algorithm 6). First, a vector $\mathbf{u}$ is generated by randomly sampling a unit vector. Then, $\mathbf{w}'$ is obtained by moving $\mathbf{w}$ by a step of size δ in the direction $\mathbf{u}$. An alternative method of candidate selection using historical data is presented in §7.1.2.

Central to the performance of DBGD is the choice of a reliable feedback mechanism. The algorithm learns using relative feedback, typically implemented in the form of an *interleaved comparison method* (i.e., a method for inferring relative comparisons between rankers). Previous to this work, DBDG was evaluated in supervised learning settings only, i.e., its effectiveness using interleaved comparison methods had not been confirmed. We give an overview of existing interleaved comparison methods in §2.3.1 and develop new ones in 4. Learning with DBGD and different interleaved comparison methods is empirically investigated in Chapters 6 and 7.

3 Problem Formulation and Experiments

In this chapter, we formalize online learning to rank for IR as a contextual bandit problem and detail our experimental setup. The problem formulation is employed in all later research chapters. The experimental setup is used to evaluate the algorithms presented in Chapters 4, 6, and 7, using learning to rank data sets and click data. A different experimental setup, based on observed user interactions with a web search engine, is used in Chapter 5, and is explained in that chapter.

Below, we first detail our problem formulation (§3.1). We then give an overview of the experimental setup (§3.2), before turning to the click models (§3.3), data sets (§3.4), and evaluation measures (§3.5) used.

3.1 Problem Formulation

We formulate the problem of online learning to rank for IR as a continuous cycle of interactions between users and a search engine, in which the search engine's goal is to provide the best possible search results at all times. In contrast to most current formulations in IR, where the search engine passively applies its ranking model, we consider it an active agent. To optimize its ranker, the search engine can learn from interaction with its users. A natural fit for this problem are formalizations from RL, in which an algorithm learns by trying out actions (e.g., document lists) that generate rewards (e.g., an evaluation measure such as NDCG) from its environment (e.g., users) (Sutton and Barto, 1998) (cf. §2.4). Using this formalization allows us to describe this problem in a principled way and to apply concepts and solutions from this well-studied area.

Figure 3.1 shows the interaction cycle. A user submits a query to a retrieval system, which generates a document list and presents it to the user. The user interacts with the list, e.g., by clicking on links, from which the retrieval system infers feedback about the quality of the presented document list. Based on the inferred feedback, the retrieval system can improve its ranker to better respond to future queries. This problem formulation directly translates to an RL problem (cf., Figure 3.1, RL terminology in italics) in which the retrieval system tries, based only on implicit feedback, to maximize a hidden reward signal that corresponds to some evaluation measure. We make the simplifying assumption that queries are independent, i.e., queries are independent of each other and

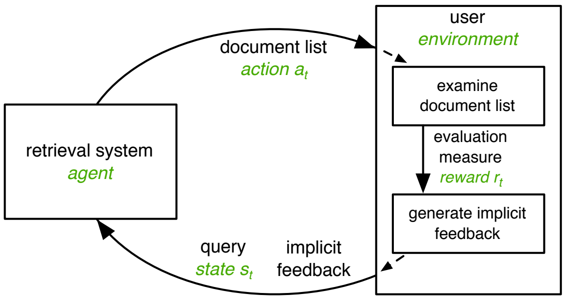


Figure 3.1: The IR problem modeled as a contextual bandit problem, with IR terminology in black and corresponding RL terminology in green and italics.

of previously displayed results.[1] This renders the problem a *contextual bandit problem* (Barto et al., 1981; Langford and Zhang, 2008) (§2.4.1).

Because our algorithms learn online, we need to measure their *online performance*, i.e., how well they address users' information needs *while learning*. Previous work in learning to rank for IR has considered only final performance, i.e., performance on unseen data after training is completed (Liu, 2009), and, in the case of active learning, learning speed in terms of the number of required training samples (Xu et al., 2010).

As is common in RL, we measure online performance in terms of *cumulative reward*, i.e., the sum of rewards over all queries addressed during learning (Sutton and Barto, 1998). Many definitions of cumulative reward are possible, depending on the modeling assumptions. We assume an *infinite horizon problem*, a formulation that is appropriate for IR learning to rank problems that run indefinitely (§2.4). As shown in Eq. 2.2, we use a discount factor $\gamma \in [0, 1)$ to weight immediate rewards higher than future rewards.

To summarize, we model online learning to rank for IR as an interaction cycle between the user and the retrieval system. We assume an infinite horizon setting and use discounting to emphasize immediate reward. This problem formulation differs from those traditionally used in IR because performance depends on cumulative reward during the entire learning process, rather than just the quality of the final retrieval system produced by learning. It also differs from typical contextual bandit problems, which assume that the agent has access to the true immediate reward resulting from its actions. Typical IR evaluation measures require explicit feedback, which is not available in most realistic use cases for online learning to rank. Thus, this contextual bandit problem is distinct in that it requires the learner to cope with implicit feedback such as click behavior (§2.3).

3.2 Experimental Setup

Evaluating the ability of an algorithm to maximize cumulative performance in an online IR setting poses unique experimental challenges. The most realistic experimental setup—in a live setting with actual users—is risky because users may get frustrated with bad

[1]This formulation corresponds to a setting where each query is submitted by a different user.

search results. The typical TREC-like setup used in supervised learning to rank for IR is not sufficient because information on user behavior is missing.

To address these challenges, we propose an evaluation setup that simulates user interactions. This setup combines data sets with explicit relevance judgments that are typically used for supervised learning to rank with recently developed click models. Given a data set with queries and explicit relevance judgments, interactions between the retrieval system and the user are simulated (cf., the box labeled "user/environment" in Figure 3.1). Submitting a query is simulated by random sampling from the set of queries. After the system has generated a result list for the query, feedback is generated using a click model and the relevance judgments provided with the data set. Note that the explicit judgments from the data set are not directly shown to the retrieval system but are used to simulate the user feedback and measure cumulative performance.

We use this evaluation setup in two scenarios, the *online evaluation scenario* and the *online learning to rank scenario*. Online evaluation is both a goal in itself and a subproblem of online learning to rank. By itself, it allows the assessment of rankers that were tuned e.g., manually, or using offline learning to rank, using real search engine traffic. As a subproblem of online learning to rank, online evaluation provides the mechanism for inferring feedback for learning.

The online evaluation scenario is investigated in Chapter 4. There, the goal of our experiments is to assess the efficiency of interleaved comparison methods when comparing different rankers, and therefore we measure how much interaction data a method needs to distinguish two rankers. We investigate the online learning to rank scenario in Chapters 6 and 7. There, we focus on online performance, i.e., we measure cumulative reward as described above. The details of each specific experiment are explained in the respective chapters as needed.

Using simulated evaluations naturally has limitations, but allows us to systematically investigate online evaluation and online learning to rank methods, without the risks associated with experiments involving real users. Here, we can show how learning methods behave under different assumptions about user behavior, but to what degree these assumptions apply in specific practical settings needs to be studied in more detail, which is beyond the scope of this thesis. We address one aspect of user behavior, caption bias, in a study of real-live search engine traffic in Chapter 5.

3.3 Click Models

Our click models are based on the Dependent Click Model (DCM) (Guo et al., 2009a,b),[2] a generalization of the cascade model (Craswell et al., 2008), that has been shown to be effective in explaining users' click behavior in web search. The model explains position bias (i.e., the observation that higher-ranked results are much more likely to be clicked than lower-ranked ones) by positing that users start examining documents at the top of a result list. For each document they examine, they determine whether the document representation (e.g., consisting of title, snippet and URL) appears promising enough to warrant a click (we model this step of deciding to click with a click probability given

[2]Models that take additional information into account have been shown to more accurately reflect click behavior (Xu et al., 2012), but these make stronger assumptions, rendering experiments unnecessarily complex.

some relevance label $P(C|R)$). After each click, users decide whether they are satisfied with the information provided in the clicked document(s) and they want to stop examining further results (with stop probability $P(S|R)$), or if they want to continue examining results.

To instantiate this click model we need to define click and stop probabilities. When the DCM is trained on large click logs, probabilities are estimated for individual query-document pairs, while marginalizing over the position at which documents were presented in the training data. In our setting, learning these probabilities directly is not possible, because no click log data is available. Therefore, we instantiate the model heuristically, making choices that allow us to study the behavior of our approach in various settings. Setting these probabilities heuristically is reasonable because learning outcomes for the gradient algorithms used in this thesis are influenced mainly by how likely users are to click on documents of different relevance grades. Thus, the ratio of these probabilities is more important than the actual numbers used to instantiate the model. We use several instantiations to cover a broad range of scenarios, from very reliable to very noisy click behavior.

We define four click models for annotated data sets with up to five relevance levels, ranging from 0 – “non-relevant” to 4 – “highly relevant”. An overview of the resulting click models is given in Table 3.1.

	click probabilities					*stop probabilities*				
relevance grade R	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	0.3	0.5	0.7	0.95	0.2	0.3	0.5	0.7	0.9
informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5
almost random	0.4	0.45	0.5	0.55	0.6	0.5	0.5	0.5	0.5	0.5

Table 3.1: Overview of the click models used in our experiments for data sets with five relevance grades. For data sets with three relevance grades, only the values for $R \in \{0, 2, 4\}$ are used. For data sets with binary relevance, only the values for the lowest ($R = 0$, non-relevant) and highest ($R = 4$, relevant) relevance grades are used.

First, to obtain an upper bound on the performance that could be obtained if feedback was very reliable, we define a *perfect click model*. This model simulates a user who clicks on all highly relevant documents ($R = 4$), and never clicks on non-relevant documents ($R = 0$). Click probabilities for intermediate relevance levels have a linear decay, except for a higher increase in click probability between relevance levels 2 and 3 (based on previous work that showed that grouping “good” documents with non-relevant documents is more effective than grouping them with relevant documents (Chapelle et al., 2009)). The stop probability for this click model is zero, meaning that there is no position bias (simulated users examine all top-10 results).

We also implement two realistic models, the *navigational* and *informational* models. These models are based on typical user behavior in web search (Broder, 2002; Guo et al., 2009a), because most of the data sets we use implement web search tasks (see below, §3.4).

The *navigational* click model simulates the focus on top-ranked and highly relevant results that are characteristic of navigational searches (Liu et al., 2006; Rose and Levinson, 2004). In a navigational task, users look for a specific document they know to exist in a collection, e.g., a company's homepage. Typically, it is easy to distinguish relevant and non-relevant documents and the probability of stopping examination after visiting a relevant result is high. Therefore, our navigational model is relatively reliable, with a strong decay in click probabilities with decreasing relevance. In comparison with the perfect click model, the navigational model results in fewer clicks on result documents, with a stronger focus on highly relevant and top-ranked results (i.e., position bias is high).

In an informational task, users look for information about a topic, which can be distributed over several pages. Here, users generally know less about what page(s) they are looking for and clicks tend to be noisier. Correspondingly, the *informational* click model captures the broader interests characteristic for informational searches (Liu et al., 2006; Rose and Levinson, 2004). In this model, the click and stop probabilities for lower relevance grades are more similar to those for highly relevant documents, resulting in more clicks, and more noisy click behavior than the previous models.

As a lower bound on click reliability, we also include an *almost random* click model, with only a small linear decay in the click probabilities for different relevance grades. This model has a strong position bias, with stop probability $P(S) = 0.5$ for all relevance grades.

3.4 Data Sets

We conduct our experiments using several standard data sets for learning to rank in IR. In Chapter 4 we use the MSLR-WEB30k data set, and in Chapters 6 and 7 we use the data sets contained in the LETOR 3.0 and 4.0 collections. All data sets consist of a set of query-document pairs, represented by up to 136 ranking features, and relevance judgments provided by professional annotators.

The MSLR-WEB30k Microsoft learning to rank data set[3] is used in our online evaluation experiments in Chapter 4. This data set was constructed to provide access to realistic training data as it is used by search engines, such as Bing. It encodes relations between queries and candidate documents in 136 precomputed features, including scores for language modeling, BM25, TF-IDF, and other retrieval models computed on different document fields (title, anchor, etc.), quality indicators, click-based features, and PageRank and other link-based features. Relevance judgments were obtained from professional web-search judges, and are provided on a 5-point scale. Our experiments use the training set of fold 1. This set contains $18,919$ queries, with an average of 9.6 (judged) candidate documents per query.

The online learning to rank experiments in Chapters 6 and 7 use the LETOR 3.0 and LETOR 4.0 collections (Liu et al., 2007). In total, these two collections comprise nine data sets.

The following search tasks are implemented: the data set *OHSUMED* models a literature search task, based on a query log of a search engine for the MedLine abstract

[3]`http://research.microsoft.com/en-us/projects/mslr/default.aspx`, retrieved on December 29, 2012.

database. This data set contains 106 queries that implement an informational search task, and about 150 judged documents per query. The remaining eight data sets are based on TREC Web track tasks run between 2003 and 2008. The data sets *HP2003*, *HP2004*, *NP2003*, and *NP2004* implement navigational tasks, homepage finding and named-page finding. *TD2003* and *TD2004* implement an informational task: topic distillation. All six data sets are based on the .GOV document collection, a crawl of the .gov domain, and contain between 50 and 150 queries and approximately 1000 judged documents per query. A more recent document collection, .GOV2 formed the basis of *MQ2007* and *MQ2008*. These data sets contain 1700 and 800 queries respectively, but far fewer judged documents per query than the other LETOR data sets (approximately 40 and 20).

The data sets *OHSUMED*, *MQ2007* and *MQ2008* are annotated with graded relevance judgments (3 grades, from 0, not relevant, to 2, highly relevant), while the remaining LETOR data sets are labeled using binary assessments. Each data set comes split up for supervised learning to rank experiments using 5-fold cross-validation. We use the training sets for training during the learning cycle and for calculating online performance, and the test sets for measuring final performance. This setup replicates the standard established for the supervised learning setting as much as possible.

3.5 Evaluation Measures

Our assessment of online evaluation and online learning to rank methods is based on NDCG, as defined in Eq. 2.1. For our online evaluation experiments (Chapters 4 and 5), we compare the outcomes of interleaved comparison methods to the true NDCG difference between rankers, following previous work (Radlinski and Craswell, 2010).

To measure online performance in online learning to rank experiments (Chapters 6 and 7), we instantiate reward as the NDCG of the (interleaved) result lists presented to the user. We then define online performance as the discounted sum of NDCG that the retrieval system accrues throughout the length of the experiment, as shown in Eq. 2.2.

Because our problem formulation assumes an infinite horizon, online performance is defined as an infinite sum of discounted rewards (cf. §3.2). Since experiments are necessarily finite, we cannot compute this infinite sum exactly. However, because the sum is discounted, rewards in the far future have little impact and cumulative performance can be approximated with a sufficiently long finite experiment.

In our experiments, we set the discount factor $\gamma = 0.995$. This choice can be justified in two ways. First, it is typical of discount factors used when evaluating RL methods (Sutton and Barto, 1998). Choosing a value close to 1 ensures that future rewards have significant weight and thus the system must explore in order to perform well. Second, at this value of γ, cumulative performance can be accurately estimated with the number of queries in our data sets. Using this discount factor, rewards after 1,000 iterations have a weight of 1% or less. Therefore, finite runs over 1,000 are good approximations of true cumulative performance.

While our main metric is online performance, we additionally report final performance (after learning) for analysis. We measure final performance as the NDCG on the held-out test set.

We test for statistically significant differences between baseline and experiment runs using a two-sided student's t-test. Unless noted otherwise, we consider differences between runs statistically significant if the obtained p-value is less than 0.05. In tables, we mark significant increases using $^\vartriangle$ ($p < 0.05$) or $^\blacktriangle$ ($p < 0.01$) and significant decreases using $^\triangledown$ ($p < 0.05$) or $^\blacktriangledown$ ($p < 0.01$). Additionally, best results per table row are highlighted in **bold** when applicable.

When results are reported in graphs, we provide 95% confidence intervals. We determine whether differences are statistically significant based on the overlap between confidence intervals.

4 Probabilistic Interleaving

Having formulated online learning to rank for IR as a contextual bandit problem (§3.1), we now turn to one component of this problem, namely how to infer reliable feedback from user clicks. In contrast to supervised learning approaches, an online learner is not provided with an appropriate set of training data, but has to elicit feedback useful for learning by interacting with the user. Because previous work found that relative interpretations of user actions are most reliable, we focus on such interpretations, in particular interleaved comparison methods (§ 2.3.1). These methods were found to be able to accurately compare the quality of competing rankers using click data, and are applicable beyond online learning to rank, to ranker evaluation in general.

This chapter addresses our research questions **RQ 1**-**RQ 7**, as specified in §1.1. We propose a framework for analyzing interleaved comparison methods, and a new set of interleaved comparison methods based on a probabilistic interpretation of the interleaving process.

First, we focus on the theoretical foundations of interleaved comparison methods, and address the following question:

RQ 1 What criteria should an interleaved comparison method satisfy to enable reliable online learning to rank for IR?

We propose to characterize these methods in terms of fidelity, soundness, and efficiency. An interleaved comparison method has *fidelity* if it measures the right quantity, i.e., if the outcome of each ranker comparison is defined such that the expected outcome properly corresponds to the true relevance of the ranked documents. It is *sound* if the estimates it computes of that expected outcome have two desirable statistical properties: namely they are unbiased and consistent. It is *efficient* if the accuracy of those estimates improves quickly as more comparisons are added.

We use the proposed framework to analyze the existing interleaved comparison methods: BI (Joachims, 2003), TD (Radlinski et al., 2008b), and DC (He et al., 2009):

RQ 2 Do current interleaved comparison methods satisfy these criteria?

We find that, although sound, none of these methods meet our criteria for fidelity. To overcome this limitation, we propose a new interleaved comparison method, *probabilistic interleave* (PI). PI is based on a probabilistic interpretation of the interleaving process, which enables it to compute comparison outcomes that are weighted by the rank differences of clicked documents. PI in its most naive form can be inefficient, because the probabilistic approach can introduce more noise than existing interleaving methods. Therefore, we derive an extension to PI that exploits the insight that probability distributions are known for some of the variables in the graphical model that describes its interleaving process. This allows us to derive a variant of PI, PI-MA, whose estimator marginalizes out these known variables, instead of relying on noisy samples of them. Regarding these new methods, we address the following questions:

RQ 3 Do PI and its extension PI-MA exhibit fidelity and soundness?

RQ 4 Is PI-MA more efficient than previous interleaved comparison methods? Is it more efficient than PI?

We prove that both PI and PI-MA have fidelity and are statistically sound, and we empirically show that PI-MA is more efficient than previous interleaved comparison methods and PI.

We also derive a second extension to PI that broadens the applicability of interleaved comparison methods by enabling them to reuse previously observed, *historical*, interaction data. Current interleaved comparison methods are limited to settings with access to *live* data, i.e., where data is gathered during the evaluation itself. Without the ability to estimate comparison outcomes using historical data, the practical utility of interleaved comparison methods is limited. If all comparisons are done with live data, then applications such as learning to rank, which perform many comparisons, need prohibitive amounts of data (cf., Chapter 7). Since interleaving result lists may affect the users' experience of a search engine, collecting live data is complicated by the need to first control the quality of the compared rankers using alternative evaluation setups. Unlike existing methods, the probabilistic nature of PI enables the use of *importance sampling* to properly incorporate historical data.

RQ 5 Can historical data be reused to compare new ranker pairs?

RQ 6 Does PI-MA-IS maintain fidelity and soundness?

RQ 7 Can PI-MA-IS reuse historical data effectively?

We prove that PI-MA-IS preserves fidelity and soundness, and empirically show that it can effectively use historical data for new ranker comparisons.

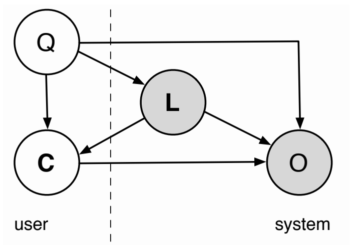
The remainder of this chapter is organized as follows. We detail our criteria for analyzing interleaved comparison methods and analyze existing methods in §4.1. In §4.2, we detail our proposed method, PI, and the two extensions PI-MA and PI-MA-IS. Our experiments for investigating the efficiency of the existing and proposed interleaved comparison methods are presented in §4.3. We detail and discuss our results in §4.4 and conclude in §4.5.

4.1 Analysis

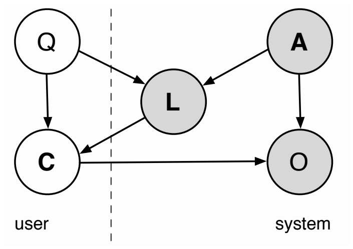
We analyze interleaved comparison methods using a probabilistic framework, and three criteria – fidelity, soundness, and efficiency – that are formulated on the basis of this framework. In this section, we first introduce our probabilistic framework and show how it relates to existing interleaved comparison methods (§4.1.1). Next, we formally define our criteria for analyzing interleaved comparison methods (§4.1.2). Finally, we use these criteria to analyze the existing interleaved comparison methods (§4.1.3–§4.1.5). The interleaved comparison methods analyzed in this section (BI, TD, and DC) have been reviewed in §2.3.1.

4.1.1 Framework

The framework we propose in this section is designed for the systematic assessment of interleaved comparison methods. In our framework, interleaved comparison methods are described probabilistically using graphical models, as shown in Figure 4.1. These models specify how a retrieval system interacts with its users and how observations from such interactions are used to compare rankers. Generally, an interleaved comparison method is completely specified by the components shown in gray, in the "system" part of the model. Figure 4.1(a) shows one variant of the model, used for BI and DC, and Figure 4.1(b) shows another, used for TD and PI (PI is introduced in §4.2 below).



(a) Graphical model for BI and DC



(b) Graphical model for TD and PI

Figure 4.1: Probabilistic model for comparing rankers (a) using BI and DC, and (b) using TD and PI. Conditional probability tables are known only for the variables in gray.

Both variants include the four random variables Q, $\mathbf{L}$, $\mathbf{C}$, and O. The interaction begins when the user submits a query $q \sim P(Q)$ to the system. We assume that $P(Q)$, though unknown to the system, is static and independent of its actions. Based on q, a result list $\mathbf{l} \sim P(\mathbf{L})$ is generated and presented to the user. Because we deal with interleaving methods, we assume that $\mathbf{l}$ is an interleaved list that combines documents obtained from the two rankers $\mathbf{l}_1(q)$ and $\mathbf{l}_2(q)$. Thus, given q, an interleaving method completely defines $P(\mathbf{L})$ (e.g., Algorithm 1, lines 1–12). The interleaved list $\mathbf{l}$ is returned to the user, who examines it and clicks on documents that may be relevant for the given q, resulting in an observation $\mathbf{c} \sim P(\mathbf{C})$ that is returned to the system. The system then uses $\mathbf{c}$, and possibly additional information, to infer a comparison outcome $o \sim P(O)$. O, which is specified by the comparison step of the method (e.g., Algorithm 1, lines 13–15), is a deterministic function of the other variables but is modeled as a random variable to simplify our analysis.

The optional components defined in the model are the dependencies of O on Q and $\mathbf{L}$ for BI and DC (cf., Figure 4.1(a)), and the assignments $\mathbf{A}$ for TD and PI (cf., Figure 4.1(b)). As shown in Algorithms 1 (page 20) and 3 (page 22), BI and DC compute outcomes using the observed $\mathbf{c}$, $\mathbf{l}$, and q (specifically, the $\mathbf{l}_1$ and $\mathbf{l}_2$ generated for that q). In contrast, the comparison function of TD (and of PI, as we will see in §4.2) does not require $\mathbf{l}$ and q, but rather uses assignments $\mathbf{a} \sim P(\mathbf{A})$ that indicate to which original ranking function the documents in $\mathbf{l}$ are assigned (cf., Algorithm 2 on page 21).

The random variables in the model have the following sample spaces. For Q, it is the (possibly infinite) universe of queries, e.g., $q =$ '*facebook*'. For $\mathbf{L}$ it is all permutations of documents, e.g., $\mathbf{l} = [d_1, d_2, d_3, d_4]$. For $\mathbf{C}$ it is all possible click vectors, such that $\mathbf{c}[i]$ is a binary value that indicates whether the document $\mathbf{l}[i]$ was clicked, e.g., $\mathbf{c} = [1, 0, 0, 0]$. For $\mathbf{A}$ it is all possible assignment vectors, such that $\mathbf{a}[i]$ is a binary value that indicates which ranker contributed $\mathbf{l}[i]$, e.g., $\mathbf{a} = [1, 2, 1, 2]$.

Within this framework, we are particularly interested in the sign of the expected outcome $E[O]$. However, $E[O]$ cannot be determined directly because it depends on the unknown Q and $\mathbf{C}$. Instead, it is estimated from sample data, using an estimator $\hat{E}[O]$. The sign of $\hat{E}[O]$ is then interpreted as follows. An $\hat{E}[O] < 0$ corresponds to inferring a preference for ranker $\mathbf{l}_1$, $\hat{E}[O] = 0$ is interpreted as a tie, and $\hat{E}[O] > 0$ is interpreted as a preference for ranker $\mathbf{l}_2$.

The simplest estimator of an expected value is the mean computed from a sample of i.i.d. observations of that value. Thus, the expected outcome can be estimated by the mean of n observed outcomes o_i:

$$\hat{E}[O] = \frac{1}{n} \sum_{i=0}^{n} o_i. \tag{4.1}$$

Previous work did not formulate estimated interleaved comparison outcomes in terms of a probabilistic framework as done here. We show below that a commonly used previous estimator is equivalent to the sample mean. Chapelle et al. (2012) formulate the following estimator:

$$\hat{E}_{wins} = \frac{wins(\mathbf{l}_2) + \frac{1}{2} ties(\mathbf{l}_{1,2})}{wins(\mathbf{l}_2) + wins(\mathbf{l}_1) + ties(\mathbf{l}_{1,2})} - 0.5. \tag{4.2}$$

Here, $wins(\mathbf{l}_i)$ denotes the number of samples for which $\mathbf{l}_i$ won the comparison, and $ties(\cdot)$ denotes the number of samples for which the two competing rankers tied. The following theorem states that this estimator is equal to the rescaled sampled mean.

Theorem 4.1.1. *The estimator in Eq. 4.2 is equal to two times the sample mean (Eq. 4.1).*

Proof. See Appendix 4.A. □

Clearly, this theorem implies that Eq. 4.2 always has the same sign as the sample mean, and thus the same preferences will be inferred.

Alternative estimators have been proposed and investigated in (Chapelle et al., 2012; Radlinski and Craswell, 2010; Yue et al., 2010a). Typically, these alternatives are designed to converge faster at the expense of obtaining biased estimates. This introduces a bias-variance trade-off. A formal analysis of these is beyond the scope of this thesis.

4.1.2 Definitions of Fidelity, Soundness, and Efficiency

Based on the probabilistic framework introduced in the previous subsection, we define our criteria for analyzing interleaved comparison methods: *fidelity*, *soundness*, and *efficiency*. These criteria reflect what interleaved comparison outcomes measure, whether an estimator of that outcome is statistically sound, and how efficiently it uses data samples. These assessment criteria are not intended to be complete, but are considered minimal requirements. Nevertheless, they enable a more systematic analysis of interleaved comparison methods than was previously attempted.

Our first criterion, fidelity, concerns whether an interleaved comparison method measures the right quantity, i.e., if $E[O|q]$ properly corresponds to the true quality difference between $\mathbf{l}_1$ and $\mathbf{l}_2$ in terms of how they rank relevant documents for a given q. Our definition uses the following concepts:

- $random\_clicks$ indicates that, for a given query, clicks are uniformly random, i.e., all documents at all ranks are equally likely to be clicked:

  $$random\_clicks(q) \Leftrightarrow \forall d_{i,j} \in \mathbf{l}, P(\mathbf{c}[r(d_i, \mathbf{l})]|q) = P(\mathbf{c}[r(d_j, \mathbf{l})]|q),$$

  where $P(\mathbf{c}[r(d_i, \mathbf{l})]|q)$ is the probability of a click at the rank r at which document d_i is displayed in result list $\mathbf{l}$.

- $correlated\_clicks(q)$ indicates positive correlation between clicks and document relevance:

  $$\begin{aligned} &correlated\_clicks(q) \Leftrightarrow \\ &\qquad \forall r \in ranks(\mathbf{l}), P(\mathbf{c}[r]|rel(\mathbf{l}[r], q)) > P(\mathbf{c}[r]|\neg rel(\mathbf{l}[r], q)), \end{aligned}$$

  where $rel(\mathbf{l}[r], q)$ indicates that the document at rank r of list $\mathbf{l}$ is relevant to q, and $P(\mathbf{c}[r]|rel(\mathbf{l}[r], q))$ is the probability of a click on the document at r, given that it is relevant. This means that, for a given query and at equal ranks, a relevant document is more likely to be clicked than a non-relevant one.

- $pareto\_dominates$ indicates that ranker $\mathbf{l}_1$ Pareto dominates $\mathbf{l}_2$ for query q:

  $$\begin{aligned} pareto\_dominates(\mathbf{l}_1, \mathbf{l}_2, q) \Leftrightarrow &\forall d \in rel(\mathbf{l}_1 \cup \mathbf{l}_2, q), r(d, \mathbf{l}_1) \geq r(d, \mathbf{l}_2) \\ &\wedge \exists d \in rel(\mathbf{l}_1 \cup \mathbf{l}_2, q), r(d, \mathbf{l}_1) > r(d, \mathbf{l}_2), \end{aligned}$$

  where $rel(\mathbf{l}_1 \cup \mathbf{l}_2, q)$ denotes the set of documents in $\mathbf{l}_1$ and $\mathbf{l}_2$ that are relevant to q. Thus, one ranker Pareto dominates another in terms of how it ranks relevant documents if and only if it ranks all relevant documents at least as high as, and at least one relevant document higher than, the other ranker.

Definition 4.1.2 (Fidelity). *An interleaved comparison method exhibits* fidelity *if,*

1. *under random clicks, the rankers tie in expectation over clicks, i.e.,*

   $$\forall q(random\_clicks(q) \Rightarrow E[O|q] = 0),$$

2. *under correlated clicks, ranker* $\mathbf{l}_2$ *is preferred if it Pareto dominates* $\mathbf{l}_1$*:*

$$\forall q (pareto\_dominates(\mathbf{l}_2, \mathbf{l}_1, q) \Rightarrow E[O|q] > 0).$$

We formulate fidelity in terms of the expected outcome for a given q because, in practice, a ranking function can be preferred for some queries and not for others. We consider the expectation over some population of queries in our definition of soundness below. In addition, we formulate condition (2) in terms of detecting a preference for $\mathbf{l}_2$. This is without loss of generality, as switching $\mathbf{l}_1$ and $\mathbf{l}_2$ results in a sign change of $E[O|q]$.

The first condition of our definition of fidelity has been previously proposed in (Radlinski et al., 2008b) and (Chapelle et al., 2012), and was used to analyze BI. A method that violates (1) is problematic because noise in click feedback can affect the outcome inferred by such a method. However, this condition is not sufficient for assessing interleaved comparison methods because a method that picks a preferred ranker at random would satisfy it, but cannot effectively infer preferences between rankers.

We add the second condition to require that an interleaved comparison method prefers a ranker that ranks relevant documents higher than its competitor. A method that violates (2) is problematic because it may fail to detect quality differences between rankers. This condition includes the assumption that clicks are positively correlated with relevance and rank. This assumption, which is implicit in previous definitions of interleaved comparison methods, is a minimal requirement for using clicks for evaluation.

Our definition of fidelity is stated in terms of binary relevance, as opposed to graded relevance, because requirements about how ranks of documents with different relevance grades should be weighted depend on the context in which an IR system is used (e.g., is a ranking with one highly relevant document better than one with three moderately relevant documents?). In addition, our definition imposes no preferences on rankings for which none dominates the other (e.g., one ranking placed relevant documents at ranks 1 and 7, the other places the same documents at ranks 3 and 4 — which is better again depends on the search setting). Because it is based on Pareto dominance, the second condition of our definition imposes only a partial ordering on ranked lists. This partial ordering is stronger than the requirements posed in previous work, with a minimal set of additional assumptions. Note that in past and present experimental evaluations, stronger assumptions are implicitly made, e.g., by using NDCG as a performance measure.

In contrast to fidelity, which focuses on outcomes for individual observations, our second criterion focuses on the characteristics of interleaved comparison methods when estimating comparison outcomes from sample data (of size n). Soundness concerns whether an interleaved comparison method's estimates of $E[O]$ are statistically sound.

Definition 4.1.3 (Soundness)**.** *An interleaved comparison method exhibits* soundness *for a given definition of O if its corresponding* $\hat{E}[O]$*, computed from sample data, is an unbiased and consistent estimator of* $E[O]$.

An estimator is *unbiased* if its expected value is equal to $E[O]$ (Halmos, 1946). It is *consistent* if it converges with probability 1 to $E[O]$ in the limit as $n \to \infty$ (Lehmann, 1999). A simple example of an unbiased and consistent estimator of the expected value of a random variable X, distributed according to some distribution $P(X)$, is the mean of samples drawn i.i.d. from $P(X)$.

Soundness has not been explicitly addressed in previous work on interleaved comparison methods. However, as shown above (§4.1.1, Theorem 4.1.1) a typical estimator proposed in previous work can be reduced to the sample mean, which is trivially sound. Soundness is more difficult to establish for some variants of our PI method introduced in §4.2, because they ignore parts of observed samples, marginalizing over known parts of the distribution in order to reduce variance. We prove in §4.2 that these variants preserve soundness.

Note that methods can perform well in practice in many cases even if they are biased, because there usually is a trade-off between bias and variance. However, all else being equal, an unbiased estimator provides more accurate estimates.

The third criterion, efficiency, concerns the amount of sample data a method requires to make reliable preference decisions.

Definition 4.1.4 (Efficiency). *Let $\hat{E}_1[O]$, $\hat{E}_2[O]$ be two estimators of expected interleaved comparison outcomes $E[O]$. $\hat{E}_1[O]$ is a more* efficient *estimator of $E[O]$ than $\hat{E}_2[O]$ if $\hat{E}_1[O]$ Pareto dominates $\hat{E}_2[O]$ in terms of accuracy for a given sample size, i.e., $\hat{E}_1[O]$ is more efficient than $\hat{E}_2[O]$ if and only if*

$$\begin{aligned}
&\forall n(P(sign(\hat{E}_1^n[O]) = sign(E[O])) \geq P(sign(\hat{E}_2^n[O]) = sign(E[O]))) \\
\wedge\ &\exists n(P(sign(\hat{E}_1^n[O]) = sign(E[O])) > P(sign(\hat{E}_2^n[O]) = sign(E[O]))),
\end{aligned}$$

where $\hat{E}_i^n[O]$ is the outcome estimated by $\hat{E}_i$ given sample data of size n.

Some interleaving methods may be more efficient than others in specific scenarios (e.g., known-item search (He et al., 2009)). However, more generally, efficiency is affected by the variance of comparison outcomes under a comparison method, and trends in efficiency can be observed when applying these methods to a large number of ranker comparisons. Here, we assess the efficiency of interleaved comparison methods experimentally, on a large number of ranker comparisons under various conditions (e.g., noise in user feedback) in §4.4.

Efficiency (also called *cost* by He et al. (2009)), has been previously proposed as an assessment criterion, and has been investigated experimentally on synthetic data (He et al., 2009) and on large-scale comparisons of individual ranker pairs in real-life web search traffic (Chapelle et al., 2012).

In addition to improving efficiency by reducing variance, subsequent interleaved comparisons can be made more efficient by reusing historical data. For methods that do not reuse historical data, the required amount of live data is necessarily linear in the number of ranker pairs to be compared. A key result of this chapter is that this requirement can be made sub-linear by reusing historical data. In the rest of this section, we include an analysis in terms of whether historical data reuse and the resulting increase in efficiency is possible for existing methods.

Below, we analyze the fidelity, soundness, efficiency, and possibility of data reuse of the existing interleaved comparison methods, BI (§4.1.3), TD (§4.1.4), and DC (§4.1.5).

4.1.3 Balanced Interleave

Fidelity. BI was previously analyzed by Radlinski et al. (2008b) and Chapelle et al. (2012). The method was shown to violate requirement (1) of fidelity. Here, we extend this argument, and provide example cases in which this violation of requirement (1) is particularly problematic. The identified problem is illustrated in Figure 4.2. Given $\mathbf{l}_1$ and $\mathbf{l}_2$ as shown, two interleaved lists can result from interleaving. The first is identical to $\mathbf{l}_1$, the second switches documents d_1 and d_2. Consider a user that randomly clicks on one of the result documents, so that each document is equally likely to be clicked. Because d_1 is ranked higher by $\mathbf{l}_1$ than by $\mathbf{l}_2$, $\mathbf{l}_1$ wins the comparison for clicks on d_1. However, $\mathbf{l}_2$ wins in all other cases, which means that it wins in expectation over possible interleaved lists and clicks. This argument can easily be extended to all possible click configurations using truth tables.

BI violates condition (1) of fidelity when, between the compared rankers, individual documents are moved up or down by more than one rank. In practice, it is possible that the direction of such ranking changes can be approximately balanced between rankers when a large number of queries are considered. However, this is unlikely in settings where the compared lists are systematically similar to each other. For example, re-ranking approaches such as (Xue et al., 2004) combine two or more ranking features. Imagine two instances of such an algorithm, where one places a slightly higher weight on one of the features than the other instance. The two rankings will be similar, except for individual documents with specific feature values, which will be boosted to higher ranks. If users were to only click a single document, the new ranker would win BI comparisons for clicks on all boosted documents (as it ranks them higher), and lose for clicks on all other documents below the first boosted document (as these are in the original order and necessarily ranked lower by the new ranker). Thus, under random clicks, the direction of preference would be determined solely by the number and absolute rank differences of boosted documents. A similar effect (in the opposite direction) would be observed for algorithms that remove or demote documents, e.g., in (near-)duplicate detection (Radlinski et al., 2011).

In addition, BI violates condition (2) of fidelity when more than one document is relevant. The reason is that only the lowest-ranked clicked document (k) is taken into account when calculating click score differences. If for both original lists the lowest-ranked clicked document has the same rank, the comparison results in a tie, even if large ranking differences exist for higher-ranked documents. Condition (2) is not violated when only one relevant document is present.

Soundness. Soundness of BI has not been explicitly investigated in previous work. However, as shown in §4.1.1, it is trivially sound because its estimator reduces to the sample mean.

Efficiency. The efficiency of BI was found to be sufficient for practical applications in (Chapelle et al., 2012). For example, several thousand impressions were required for detecting ranker changes that are typical for incremental improvements at commercial search engines with high confidence.

Data Reuse. Reusing historical data to compare new target rankers using BI is possible in principle. Given historical result lists and clicks, and a new pair of target rankers,

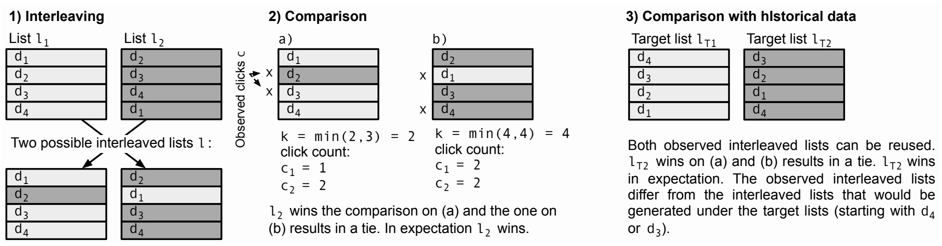

1) Interleaving
List l1
List l2
d1 d2 d3 d4
d2 d3 d4 d1
Two possible interleaved lists l:
d1 d2 d3 d4
d2 d1 d3 d4
2) Comparison
a)
b)
Observed clicks c
x
k = min(2,3) = 2
click count:
c1 = 1
c2 = 2
k = min(4,4) = 4
click count:
c1 = 2
c2 = 2
l2 wins the comparison on (a) and the one on (b) results in a tie. In expectation l2 wins.
3) Comparison with historical data
Target list lT1
Target list lT2
d4 d3 d2 d1
d3 d2 d1 d4
Both observed interleaved lists can be reused. lT2 wins on (a) and (b) results in a tie. lT2 wins in expectation. The observed interleaved lists differ from the interleaved lists that would be generated under the target lists (starting with d4 or d3).

Figure 4.2: Interleaving (1) and comparison with BI using live data (2) and historical data (3).

comparison outcomes can be computed as under live data, following Algorithm 1, lines 13–17. This means that observed clicks would be projected onto the new target lists to determine k, the rank at which the lowest click would occur for the target rankers. Then, the number of clicks on the top$k$ results can be counted for the target rankers as if they had been used in a live comparison. However, such straightforward data reuse would severely bias the inferred comparison outcomes. In particular, the target ranker that is more similar to those under which the historical data was originally collected will be likely to be preferred when data is reused. It is not clear whether and how the differences between observed interleaved lists and "correct" interleaved lists for the new target rankers could be compensated for.

4.1.4 Team Draft

Fidelity. TD was designed to address fidelity requirement (1) (Radlinski et al., 2008b). This is achieved by using assignments as described in §2.3.1. That the requirement is fulfilled can be seen as follows. Each ranker is assigned the same number of documents in the interleaved result list in expectation (by design of the interleaving process). Rankers get credit for clicks if and only if they are assigned to them. Thus, if clicks are randomly distributed, each ranker is credited with the same number of clicks in expectation.

However, TD violates fidelity requirement (2) when the original lists are similar to each other. Figure 4.3 illustrates such a case. Consider the original lists $\mathbf{l}_1$ and $\mathbf{l}_2$. Also, assume that d_3 is the only relevant document, and is therefore more likely to be clicked than other documents. We can see that $\mathbf{l}_2$ ranks d_3 higher than $\mathbf{l}_1$ (i.e., $pareto\_dominates(\mathbf{l}_2, \mathbf{l}_1, q) = true$; cf. §4.1.2), and therefore $\mathbf{l}_2$ should win the comparison. When TD is applied, four possible interleaved lists can be generated, as shown in the figure. All these possible interleaved lists place document d_3 at the same rank. In two interleaved lists, d_3 is contributed by $\mathbf{l}_1$, and in two cases it is contributed by $\mathbf{l}_2$. Thus, in expectation, both lists obtain the same number of clicks for this document, yielding a tie. In the example shown, the lists would also tie if d_4 was the only relevant document, while in cases where only d_2 is relevant, a preference for $\mathbf{l}_2$ would be detected.

In practice, TD's violation of requirement (2) can result in insensitivity to small ranking changes. As shown above, some changes by one rank may result in a difference being detected while others are not detected. This is expected to be problematic in cases where a new ranking-function affects a large number of queries by a small amount, i.e., doc-

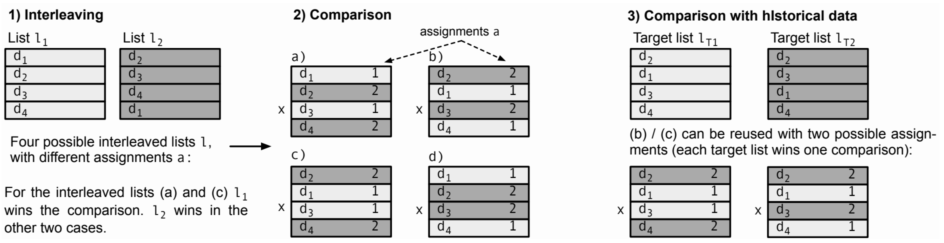

1) Interleaving
List l_1
List l_2
Four possible interleaved lists l, with different assignments a:
For the interleaved lists (a) and (c) l_1 wins the comparison. l_2 wins in the other two cases.
2) Comparison
assignments a
3) Comparison with historical data
Target list l_{T1}
Target list l_{T2}
(b) / (c) can be reused with two possible assignments (each target list wins one comparison):

Figure 4.3: Interleaving (1) and comparison with *team draft* using live data (2) and historical data (3).

uments are moved up or down by one rank, as only some of these changes would be detected. In addition, it can result in a loss of efficiency, because, when some ranking differences are not detected, more data is required to reliably detect differences between rankers.

Soundness. As with BI, the soundness of TD has not been analyzed in practice. However, as above, typical estimators produce estimates that can easily be rescaled to the sample mean, which is consistent and unbiased (cf., Theorem 4.1.1). Building on TD, methods that take additional sources of information into account have been proposed to increase the efficiency of interleaved comparisons (Chapelle et al., 2012; Yue et al., 2010a). The resulting increase in efficiency may come at the expense of soundness. A detailed analysis of these extensions is beyond the scope of this thesis.

Efficiency. As with BI, the efficiency of TD was found to be sufficient for practical applications in web and literature search (Chapelle et al., 2012). The amount of sample data required was within the same order of magnitude as for BI, with TD requiring slightly fewer samples in some cases and vice versa in others. In an analysis based on synthetic data, TD was found to be less efficient than BI on a simulated known-item search task (i.e., searches with only one relevant document) (He et al., 2009). This result is likely due to TD's lack of sensitivity under small ranking changes.

Data Reuse. Reusing historical data under TD is difficult due to the use of assignments. One option is to use only observed interleaved lists that could have been constructed under the target rankers for the historical query. If the observed interleaved lists can be generated with the target rankers, the assignment under which this would be possible can be used to compute comparison outcomes. If several assignments are possible, one can be selected at random, or outcomes for all possible assignments can be averaged. An example is shown in Figure 4.3. Given the observed interleaved lists shown in step (2), and two target rankers $\mathbf{l}_{T1}$ and $\mathbf{l}_{T2}$, the observed document rankings (b) and (c) could be reused, as they are identical to lists that can be produced under the target rankers. However, this approach is extremely inefficient. If we were to obtain historical data under a ranker that presents uniformly random permutations of candidate documents to users, of the $d!$ possible orderings of d documents that could be observed, only an expected $2^{\frac{d}{2}}$ could actually be used for a particular pair of target rankers. Even for a shallow pool of 10 candidate documents per query, these figures differ by five orders of magnitude.

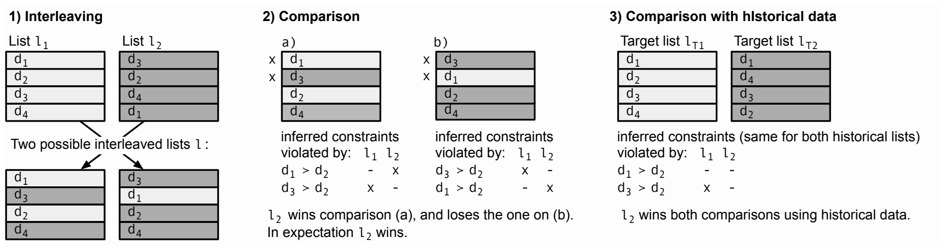


Figure 4.4: Interleaving (1) and comparison with *document constraints* using live data (2) and historical data (3).

In typical settings, where candidate pools can be large, a prohibitively large amount of data would have to be collected and only a tiny fraction of it could be reused. Thus, the effectiveness of applying TD to historical data depends on the similarity of the document lists under the original and target rankers, but is generally expected to be very low.

Even in cases where data reuse is possible because ranker pairs are similar, TD may violate requirement (2) of fidelity under historical data. An example that is analogous to that under live data is shown in Figure 4.3. Here, the lists would tie in the case that document d_3 is relevant, even though $\mathbf{l}_{T2}$ Pareto dominates $\mathbf{l}_{T1}$. In addition, reusing historical data under TD affects soundness because only some of the interleaved lists that are possible under the target rankers may be found in observed historical data. For example, in Figure 4.3, only interleaved lists that place d_2 at the top rank match the observed data and not all possible assignments can be observed. In this example, clicks on d_2 would result in wins for $\mathbf{l}_{T2}$, although the target lists place this document at the same rank. This problem can be considered a form of sampling bias, but it is not clear how it can be corrected for.

4.1.5 Document Constraints

Fidelity. DC has not been previously analyzed in terms of fidelity. We find that DC violates both requirements (1) and (2). An example is provided in Figure 4.4. The original lists $\mathbf{l}_1$ and $\mathbf{l}_2$, and the possible interleaved lists are shown. In the example, $\mathbf{l}_2$ wins in expectation, because it is less similar to the possible interleaved lists and can therefore violate fewer constraints inferred from clicks on these lists. For example, consider the possible constraints that d_1 (ranked higher by $\mathbf{l}_1$) and d_4 (ranked higher by $\mathbf{l}_2$) can be involved in. Clicks on the possible interleaved lists could result in 14 constraints that prefer other documents over d_4, but in 24 constraints that prefer other documents over d_1. As a result, $\mathbf{l}_1$ violates more constraints in expectation, and $\mathbf{l}_2$ wins the comparison in expectation under random clicks.

The example above also violates requirement (2). Consider two relevant documents, d_1 and d_3 are clicked by the user. In this case, $\mathbf{l}_1$ should win the comparison as it Pareto dominates $\mathbf{l}_2$. However, for the interleaved lists generated for this case, each original list violates exactly one constraint, which results in a tie. The reason for the violation of both requirements of fidelity is that the number of requirements each list and each document is involved in is not controlled for. It is not clear whether and how controlling for the number of constraints is possible when making comparisons using DC.

Soundness. As with BI and TD, soundness of the DC estimator can be easily established, as it is based on the sample mean (Theorem 4.1.1).

Efficiency. The efficiency of DC was previously studied on synthetic data (He et al., 2009). On the investigated cases (known-item search, easy and hard high-recall tasks with perfect click feedback), DC was demonstrated to be more efficient than BC and TD. DC has not been evaluated in a real live application.

Data Reuse. Finally, we consider applying DC to historical data. Doing this is in principle possible, because constraints inferred from previously observed lists can easily be compared to new target rankers. However, the fidelity of outcomes cannot be guaranteed (as under live data). An example is shown in part (3) of Figure 4.4. Two new target lists are compared using the historical data collected in earlier comparisons. Again, two documents are relevant, d_1 and d_3. The target lists place these relevant documents at the same ranks. However, $\mathbf{l}_1$ violates more constraints inferred from the historical data than $\mathbf{l}_2$, so that a preference for $\mathbf{l}_2$ is detected using either historical observation. As with live data, the number of constraints that can be violated by each original list is not controlled for. Depending on how the historical result list was constructed, this can lead to outcomes that are biased similarly or more strongly than under live data.

4.2 Probabilistic Interleave Methods

In this section, we present a new interleaved comparison method called *probabilistic interleave* (PI). We first give an overview of the interleaving algorithm and provide a naive estimator of comparison outcomes (§4.2.1). We show that this approach exhibits fidelity and soundness, but that its efficiency is expected to be low. Then, we introduce two extensions of PI, that increase efficiency while maintaining fidelity and soundness. The first extension, PI-MA, is based on marginalizing over possible comparison outcomes for observed samples (§4.2.2). The second extension, PI-MA-IS, shows how historical data can be reused to further increase efficiency (§4.2.3).

4.2.1 Probabilistic Interleave

We propose a probabilistic form of interleaving in which the interleaved document list $\mathbf{l}$ is constructed, not from fixed lists $\mathbf{l}_1$ and $\mathbf{l}_2$ for a given query q, but from *softmax* functions $s(\mathbf{l}_1)$ and $s(\mathbf{l}_2)$ that transform fixed result lists into probability distributions over documents. The use of softmax functions is key to our approach, as it ensures that every document has a non-zero probability of being selected by each ranker and for each rank of the interleaved result list. As a result, the distribution of credit accumulated for clicks is smoothed, based on the relative rank of the document in the original result lists. If both rankers place a given document at the same rank, then the corresponding softmax functions have the same probability of selecting it and thus they accumulate the same number of clicks in expectation. More importantly, rankers that put a given document at similar ranks receive similar credit in expectation. The difference between these expectations reflects the magnitude of the difference between the two rankings. In

Algorithm 7 Probabilistic Interleave.

```
1:  Input: $\mathbf{l}_1, \mathbf{l}_2, \tau$
2:  $\mathbf{l} \leftarrow []$
3:  $\mathbf{a} \leftarrow []$
4:  for $i \in (1, 2)$ do
5:      initialize $s(\mathbf{l}_i)$ using Eq. 4.3
6:  while $(\exists r : \mathbf{l}_1[r] \notin \mathbf{l}) \vee (\exists r : \mathbf{l}_2[r] \notin \mathbf{l})$ do
7:      $a \leftarrow 1$ if $random\_bit()$ else $2$
8:      $\bar{a} \leftarrow 2$ if $a = 1$ else $1$
9:      $append(\mathbf{a}, a)$
10:     $d_{next} \leftarrow sample\_without\_replacement(s(\mathbf{l}_a))$
11:     $append(\mathbf{l}, d_{next})$
12:     $remove\_and\_renormalize(s(\mathbf{l}_{\bar{a}}), d_{next})$
    // present $\mathbf{l}$ to user and observe clicks $\mathbf{c}$
13: compute $o$, e.g., using Eqs. 4.5–4.8
14: return $(o, (\mathbf{l}, \mathbf{a}, \mathbf{c}))$
```

this way, the method becomes sensitive to even small differences between rankings and can accurately estimate the magnitude of such differences.

The softmax functions $s(\mathbf{l}_1)$ and $s(\mathbf{l}_2)$ for given ranked lists $\mathbf{l}_1$ and $\mathbf{l}_2$ are generated by applying a monotonically decreasing function over document ranks, so that documents at higher ranks are assigned higher probabilities. Many softmax functions are possible, including the sigmoid or normalized exponential functions typically used in neural networks and RL (Lippmann, 2002; Sutton and Barto, 1998). Here, we use a function in which the probability of selecting a document is inversely proportional to a power of the rank $r_i(d)$ of a document d in list $\mathbf{l}_i$:

$$s(\mathbf{l}_i) := P_i(d) = \frac{\frac{1}{r_i(d)^\tau}}{\sum_{d' \in \mathcal{D}} \frac{1}{r_i(d')^\tau}}, \tag{4.3}$$

where $\mathcal{D}$ is the set of all ranked documents, including d. The denominator is a normalizing constant that ensures that the probabilities sum to 1. Because this softmax function has a steep decay at top ranks, it is suitable for an IR setting in which correctly ranking the top documents is the most important. It also has a slow decay at lower ranks, preventing underflow in calculations. The parameter τ controls how quickly selection probabilities decay as rank decreases, similar to the Boltzmann temperature in the normalized exponential function (Sutton and Barto, 1998). In relation to traditional IR metrics, τ can be interpreted as a discount factor that controls the focus on top ranked documents, similarly to, e.g., the rank discount in NDCG (Järvelin and Kekäläinen, 2002). In our experiments (§4.3), we use a default of $\tau = 3$ and explore possible choices of τ and their relation to traditional evaluation metrics.

After constructing $s(\mathbf{l}_1)$ and $s(\mathbf{l}_2)$, $\mathbf{l}$ is generated similarly to TD (cf., Algorithm 7). However, instead of randomizing the ranker to contribute the next document per pair, one of the softmax functions is randomly selected at each rank (line 7). Doing so is mathematically convenient, as the only component that changes at each rank is the distribution over documents. More importantly, this change ensures fidelity, as will be shown shortly. During interleaving, the system records which softmax function was selected to contribute the next document in assignment $\mathbf{a}$ (line 9). Then, a document is randomly sampled without replacement from the selected softmax function (line 10) and added to the interleaved list (line 11). The document is also removed from the non-sampled softmax function, and this softmax function is renormalized (line 12). This process repeats until $\mathbf{l}$ has the desired length.

The interleaved result list is then shown to the user in response to the query and user clicks $\mathbf{c}$ are observed, where each entry in $\mathbf{c}$ indicates whether the corresponding document in $\mathbf{l}$ has been clicked. The observed clicks are used by the second step, *comparison*. Comparison outcomes can be computed as under TD, i.e., by counting the clicks c_1 and c_2 assigned to each softmax function and returning $o = (-1$ **if** $c_1 > c_2$ **else** 1 **if** $c_1 < c_2$ **else** $0)$. An alternative method for computing comparison outcomes more efficiently is developed in §4.2.2. Finally, the algorithm returns both the computed outcome, and, in case the observed sample is to be reused as historical data, the generated $(\mathbf{l}, \mathbf{a}, \mathbf{c})$.

PI exhibits fidelity for the following reasons. To verify condition (1), consider that each softmax function is assigned the same number of documents to each rank in expectation (by design of the interleaving process). Clicks are credited to the assigned softmax function only, which means that in expectation the softmax functions tie under random user clicks. To verify condition (2), consider that each softmax function has a non-zero probability of contributing each document to each rank of the interleaved list. This probability is strictly higher for documents that are ranked higher in the result list underlying the softmax function, because the softmax functions are monotonically decreasing and depend on the document rank only. The softmax function that assigns a higher probability to a particular document d_x has a higher probability of contributing that document to $\mathbf{l}$, which gives it a higher probability of being assigned clicks on d_x. Thus, in expectation, the softmax function that ranks relevant documents higher obtains more clicks, and therefore has higher expected outcomes if clicks are correlated with relevance. In cases where $\mathbf{l}_1$ and $\mathbf{l}_2$ place d_x at the same rank, the softmax functions assign the same probability to that document, because the softmax functions have the same shape. Thus, for documents placed at the same rank, clicks tie in expectation.

An issue related to fidelity that has not been addressed previously is what the magnitude of differences in outcomes should be if, for example, a ranker moves a relevant document from rank 3 to 1, or from rank 7 to 5. In our definition of fidelity, this question is left open, as it requires additional assumptions about user expectations and behavior. In PI, this magnitude can be determined by the choice of softmax function. For example, when using the formulation in Eq. 4.3, rank discounts decrease as $\tau \rightarrow 0$. Rank discounts increase as $\tau \rightarrow \infty$, and probabilistic interleaving with deterministic ranking functions is the limiting case (this case is identical to changing TD so that rankers are randomized per rank instead of per pair of ranks). Interpreted in this way, we see that PI defines a class of interleaved comparison metrics that can be adapted to different scenarios.

As discussed in §4.1.2, the simplest estimator of $E[O]$ is the mean of the sample outcomes (Eq. 4.1). Since the sample mean is unbiased and consistent, soundness is trivially established. A limitation of this naive estimator is that its efficiency is expected to be low. In comparison to existing interleaved comparison methods, additional noise is introduced by the higher amount of randomization when selecting softmax functions per rank, and by using softmax functions instead of selecting documents from the contributing lists deterministically. In the next sections, we show how probabilistic interleaving allows us to derive more efficient estimators while maintaining fidelity and soundness.

4.2.2 Probabilistic Comparisons with Marginalization

In the previous subsection, we described PI and showed that it has fidelity and soundness. In this section, we introduce a more efficient estimator, PI-MA, that is derived by exploiting known parts of the probabilistic interleaving process, and show that under this more efficient estimator fidelity and soundness are maintained.

To derive PI-MA, we start by modeling PI using the graphical model in Figure 4.1(b).[1] This allows us to rewrite Eq. 4.1 as:

$$\hat{E}[O] = \frac{1}{n}\sum_{i=0}^{n} o_i = \frac{1}{n}\sum_{i=1}^{n}\sum_{o\in O} oP(o|\mathbf{a}_i, \mathbf{c}_i, \mathbf{l}_i, q_i), \tag{4.4}$$

where $\mathbf{a}_i$, $\mathbf{c}_i$ and $\mathbf{l}_i$, and q_i are the observed assignment, clicks, interleaved list, and query for the i-th sample. This formulation is equivalent to Eq., 4.1 because o is deterministic given $\mathbf{a}$ and $\mathbf{c}$.

In Eq. 4.4, the expected outcome is estimated directly from the observed samples. However, the distributions for $\mathbf{A}$ and $\mathbf{L}$ are known given q. As a result, we need not consider only the observed assignments. Instead, we can consider all possible assignments that could have co-occurred with each observed interleaved list $\mathbf{l}$, i.e., we can marginalize over all possible values of $\mathbf{A}$ for a given $\mathbf{l}_i$ and q_i. This method reduces the noise that results from randomized assignments, making it more efficient than methods that directly use observed assignments. Marginalizing over $\mathbf{A}$ leads to the alternative estimator:

$$\hat{E}[O] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{a}\in\mathbf{A}}\sum_{o\in O} oP(o|\mathbf{a}, \mathbf{c}_i)P(\mathbf{a}|\mathbf{l}_i, q_i). \tag{4.5}$$

The estimator in Eq. 4.5 marginalizes over all possible assignments that could have led to observing $\mathbf{l}$ by making use of the fact that this distribution is fully known. The probability of an assignment given observed lists and queries is computed using Bayes' rule:

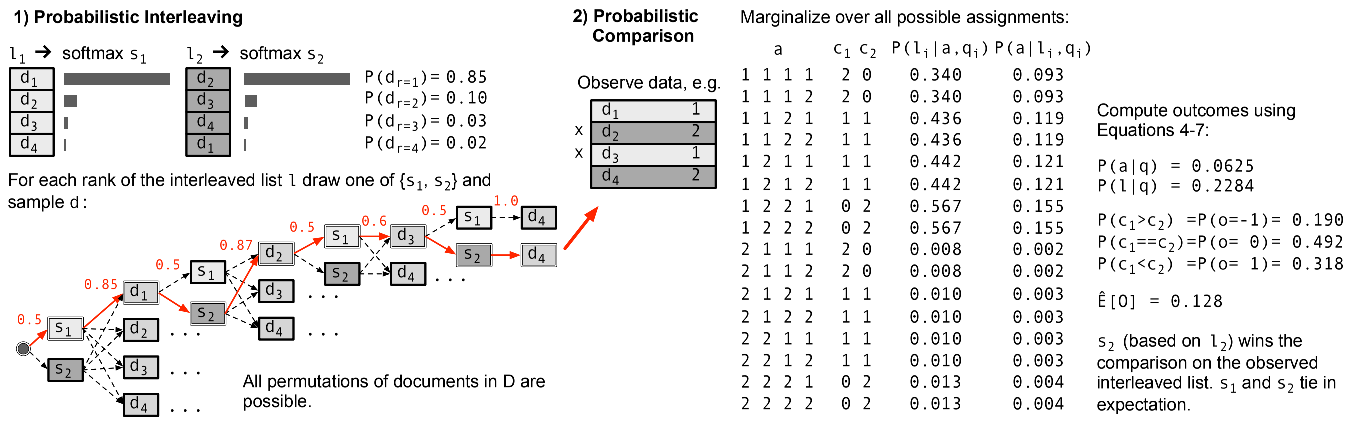
$$P(\mathbf{a}|\mathbf{l}, q) = \frac{P(\mathbf{l}|\mathbf{a}, q)P(\mathbf{a}|q)}{P(\mathbf{l}|q)}. \tag{4.6}$$

Note that $P(\mathbf{a}|q) = P(\mathbf{a}) = \frac{1}{|\mathbf{A}|}$, because $\mathbf{a}$ and q are independent. $P(\mathbf{l}|\mathbf{a}, q)$ is fully specified by the probabilistic interleaving process and can be obtained using:

$$P(\mathbf{l}|\mathbf{a}, q) = P(\mathbf{l}, \mathbf{a}|q)P(\mathbf{a}|q) = \prod_{r=1}^{len(\mathbf{l})} P(\mathbf{l}[r] \mid \mathbf{a}[r], \mathbf{l}[1, r-1], q)P(\mathbf{a}|q). \tag{4.7}$$

Here, $len(\mathbf{l})$ is the length of the document list, $\mathbf{l}[r]$ denotes the document placed at rank r in the interleaved list $\mathbf{l}$, $\mathbf{l}[1, r-1]$ contains the documents added to the list before r, and

[1] In contrast to (Hofmann et al., 2011c), we treat the outcome O as a random variable. This leads to an equivalent estimator that is more convenient for the proof in Appendix 4.B.


1) Probabilistic Interleaving
l1 → softmax s1
l2 → softmax s2
d1
d2
d3
d4
d2
d3
d4
d1
P(d_{r=1}) = 0.85
P(d_{r=2}) = 0.10
P(d_{r=3}) = 0.03
P(d_{r=4}) = 0.02
For each rank of the interleaved list l draw one of {s1, s2} and sample d:
0.5
0.85
0.5
0.87
0.5
0.6
0.5
1.0
s1
s2
All permutations of documents in D are possible.
2) Probabilistic Comparison
Observe data, e.g.
x
x
Marginalize over all possible assignments:

d	
d_1	1
x d_2	2
x d_3	1
d_4	2

Marginalize over all possible assignments:

a	c_1 c_2	$P(l_i \mid a, q_i)$	$P(a \mid l_i, q_i)$
1 1 1 1	2 0	0.340	0.093
1 1 1 2	2 0	0.340	0.093
1 1 2 1	1 1	0.436	0.119
1 1 2 2	1 1	0.436	0.119
1 2 1 1	1 1	0.442	0.121
1 2 1 2	1 1	0.442	0.121
1 2 2 1	0 2	0.567	0.155
1 2 2 2	0 2	0.567	0.155
2 1 1 1	2 0	0.008	0.002
2 1 1 2	2 0	0.008	0.002
2 1 2 1	1 1	0.010	0.003
2 1 2 2	1 1	0.010	0.003
2 2 1 1	1 1	0.010	0.003
2 2 1 2	1 1	0.010	0.003
2 2 2 1	0 2	0.013	0.004
2 2 2 2	0 2	0.013	0.004

Compute outcomes using Equations 4-7:

$P(a \mid q) = 0.0625$
$P(l \mid q) = 0.2284$

$P(c_1 > c_2) = P(o = -1) = 0.190$
$P(c_1 == c_2) = P(o = 0) = 0.492$
$P(c_1 < c_2) = P(o = 1) = 0.318$

$\hat{E}[O] = 0.128$

s_2 (based on l_2) wins the comparison on the observed interleaved list. s_1 and s_2 tie in expectation.

Figure 4.5: Example probabilistic interleaving (1) and comparison (2) with marginalization over all possible assignments.

$\mathbf{a}[r]$ denotes the assignment at r, i.e., which list contributed the document at that rank. Finally, $P(\mathbf{l}|q)$ can be computed as follows:

$$P(\mathbf{l}|q) = \sum_{\mathbf{a} \in \mathbf{A}} P(\mathbf{l}|\mathbf{a}, q) P(\mathbf{a}). \tag{4.8}$$

An example comparison using PI-MA is shown in Figure 4.5. In it, an interleaved list is generated using the process shown in Algorithm 7, in this case $\mathbf{l} = [d_1, d_2, d_3, d_4]$ (with the observed assignment $\mathbf{a} = [1, 2, 1, 2]$), as marked in red. After observing clicks on d_2 and d_3, the naive estimator detects a tie ($o = 0$), as both original lists obtain one click. In contrast, the probabilistic comparison shown in step 2 marginalizes over all possible assignments, and detects a preference for $\mathbf{l}_2$.

Next, we establish the soundness of PI-MA by showing that it is an unbiased and consistent estimator of the target outcome $E[O]$. Because PI exhibits fidelity (cf. §4.2.1), showing that PI-MA is a consistent and unbiased estimator of the same quantity establishes fidelity as well.

Theorem 4.2.1. *The following estimator is unbiased and consistent given samples from an interleaving experiment conducted according to the graphical model in Figure 4.1(b) (Eq. 4.5):*

$$\hat{E}[O] = \frac{1}{n} \sum_{i=1}^{n} \sum_{\mathbf{a} \in \mathbf{A}} \sum_{o \in O} o P(o|\mathbf{a}, \mathbf{c}_i) P(\mathbf{a}|\mathbf{l}_i, q_i).$$

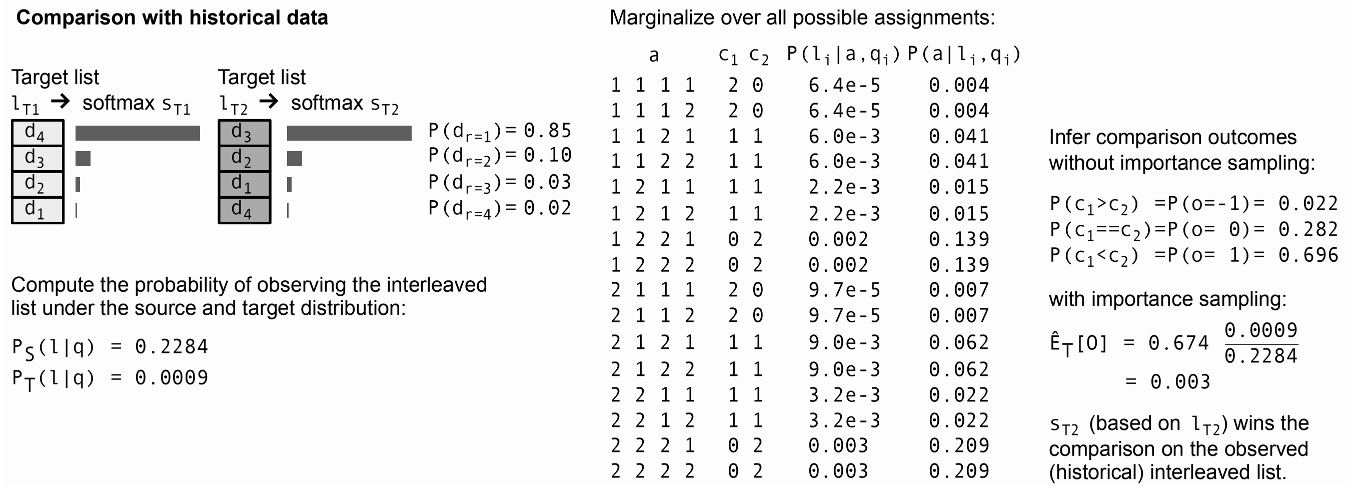
Proof. See Appendix 4.B. □

Theorem 4.2.1 establishes soundness for PI-MA (Eq. 4.5), which is designed to be more efficient than the naive estimator (Eq. 4.4). We report on an empirical evaluation of the effectiveness of these estimators in §4.4.

4.2.3 Probabilistic Comparisons with Historical Data

In the previous subsections, we derived two estimators for inferring preferences between rankers using live data. We now turn to the historical data setting, where previously collected data (e.g., from an earlier comparison of different rankers) is used to compare a new ranker pair. As shown above (cf., §4.1), none of the existing interleaved comparison methods can reuse data while maintaining fidelity and soundness. Here, we show that this is possible for a new estimator, PI-MA-IS, that we derive from PI-MA.

In principle, PI-MA, as defined in Eq. 4.5 could be directly applied to historical data. Note that, for a ranker pair that re-ranks the same set of candidate documents D as the method used to collect the historical data, $P(\mathbf{a}|\mathbf{l}, q)$ is known and non-zero for all possible assignments. Such an application of the method designed for live data could be efficient because it marginalizes over possible assignments. However, the soundness of the estimator designed for live data would be violated because the use of historical data would introduce bias, i.e., the expected outcome under historical data would not necessarily equal the expected value under live data. Similarly, the estimator would not be consistent.



Comparison with historical data

$P(d_{r=1}) = 0.85$
$P(d_{r=2}) = 0.10$
$P(d_{r=3}) = 0.03$
$P(d_{r=4}) = 0.02$

Compute the probability of observing the interleaved list under the source and target distribution:

$P_S(l|q) = 0.2284$
$P_T(l|q) = 0.0009$

Marginalize over all possible assignments:

a	c_1 c_2	$P(l_i\|a,q_i)$	$P(a\|l_i,q_i)$
1 1 1 1	2 0	6.4e-5	0.004
1 1 1 2	2 0	6.4e-5	0.004
1 1 2 1	1 1	6.0e-3	0.041
1 1 2 2	1 1	6.0e-3	0.041
1 2 1 1	1 1	2.2e-3	0.015
1 2 1 2	1 1	2.2e-3	0.015
1 2 2 1	0 2	0.002	0.139
1 2 2 2	0 2	0.002	0.139
2 1 1 1	2 0	9.7e-5	0.007
2 1 1 2	2 0	9.7e-5	0.007
2 1 2 1	1 1	9.0e-3	0.062
2 1 2 2	1 1	9.0e-3	0.062
2 2 1 1	1 1	3.2e-3	0.022
2 2 1 2	1 1	3.2e-3	0.022
2 2 2 1	0 2	0.003	0.209
2 2 2 2	0 2	0.003	0.209

Infer comparison outcomes without importance sampling:

$P(c_1>c_2) = P(o=-1) = 0.022$
$P(c_1==c_2) = P(o=0) = 0.282$
$P(c_1<c_2) = P(o=1) = 0.696$

with importance sampling:

$$\hat{E}_T[O] = 0.674 \frac{0.0009}{0.2284} = 0.003$$

s_{T2} (based on l_{T2}) wins the comparison on the observed (historical) interleaved list.

Figure 4.6: Example probabilistic comparison with historical data. We assume observed historical data as shown in Figure 4.5 above.

To see why bias and inconsistency would be introduced, consider two pairs of rankers. Pair S is the source ranker pair, which was compared in a live experiment using interleaved result lists from which the comparison outcome was computed using the resulting clicks. All data from this past experiment were recorded, and we want to compare a new ranker pair T using this historical data. Observations for pair S occur under the original distribution P_S, while observations for pair T occur under the target distribution P_T. The difference between P_S and P_T is that the two ranker pairs result in different distributions over $\mathbf{L}$. For example, interleaved lists that place documents ranked highly by the rankers in S at the top are more likely under P_S, while they may be much less likely under P_T. Bias and inconsistency would be introduced if, e.g., one of the rankers in T would be more likely to win comparisons on lists that are more likely to be observed under P_S than under P_T.

Our goal is to estimate $E_T[O]$, the expected outcome of comparing T, given data from the earlier experiment of comparing S, by compensating for the difference between P_T and P_S. To derive an unbiased and consistent estimator, note that P_T and P_S can be seen as two different instantiations of the graphical model in Figure 4.1(b). Also note that both instantiations have the same event spaces (i.e., the same queries, lists, click and assignment vectors are possible), and, more importantly, only the distributions over $\mathbf{L}$ change for different ranker pairs. Between those instantiations, the distributions over $\mathbf{A}$ are the same by design of the interleaving process. Distributions over $\mathbf{C}$ (conditioned on $\mathbf{L}$) and Q are the same for different ranker pairs, because we assume that clicks and queries are drawn from the same static distribution, independently of the ranker pair used to generate the presented list.

A naive estimator of the expected outcome $E_T[O]$ from sample data observed under P_S can be obtained from the definition of the importance sampling estimator in Eq. 2.3 with $f(\mathbf{a}, \mathbf{c}) = \sum_{o \in O} oP(o|\mathbf{a}, \mathbf{c})$:

$$\hat{E}_T[O] = \frac{1}{n} \sum_{i=1}^{n} \sum_{o \in O} oP_T(o|\mathbf{a}_i, \mathbf{c}_i) \frac{P_T(\mathbf{a}_i, \mathbf{c}_i)}{P_S(\mathbf{a}_i, \mathbf{c}_i)} \tag{4.9}$$

We refer to this estimator as PI-IS. It simply applies importance sampling to reweight observations by the ratio of their probability under the source and target distributions. Importance sampling has been shown to produce unbiased and consistent estimates of the expected outcome under the target distribution, $E_T[O]$, as long as P_S and P_T have the same event space, and P_S is non-zero for all events that have a non-zero probability under P_T (this is given by our definition of probabilistic interleaving, as long as the softmax functions under P_S are non-zero all documents that have non-zero probabilities under P_T) (MacKay, 1998). Although this estimator is unbiased and consistent, it is expected to be inefficient, because it merely reweights the original, noisy, estimates, which can lead to high overall variance.

To derive an efficient estimator of $E_T[O]$, we need to marginalize over all possible assignments, as in §4.2.2. Building on Eq. 4.9, we marginalize over the possible assignments (so the assignments $\mathbf{a}_i$ observed with the sample data are not used) and obtain the estimator PI-MA-IS:

$$\hat{E}_T[O] = \frac{1}{n} \sum_{i=1}^{n} \sum_{\mathbf{a} \in \mathbf{A}} \sum_{o \in O} oP_T(o|\mathbf{a}, \mathbf{c}_i) P_T(\mathbf{a}|\mathbf{l}_i, q_i) \frac{P_T(\mathbf{l}_i|q_i)}{P_S(\mathbf{l}_i|q_i)}. \tag{4.10}$$

As in the previous section, $P(\mathbf{a}|\mathbf{l}, q)$ is computed using Eq. 4.7, and $P(\mathbf{l}|q)$ is obtained from Eq. 4.8. An example is given in Figure 4.6. In this example, the target lists are very different from the original lists, which is reflected in the low probability of the observed interleaved list under the target distribution ($P_T(\mathbf{l}|q) = 0.0009$). Although $\mathbf{l}_{T2}$ performs much better for the observed query, the small importance weight results in only a small win for this target list.

The following theorem establishes the soundness of PI-MA-IS. By showing that Eq. 4.10 is an unbiased and consistent estimator of $E_T[O]$ under historical data, we also show that it maintains fidelity.

Theorem 4.2.2. *The following estimator is unbiased and consistent given samples from an interleaving experiment conducted according to the graphical model in Figure 4.1(b) under P_S:*

$$\hat{E}_T[O] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{a}\in\mathbf{A}}\sum_{o\in O} oP_T(o|\mathbf{a}, \mathbf{c}_i)P_T(\mathbf{a}|\mathbf{l}_i, q_i)\frac{P_T(\mathbf{l}_i|q_i)}{P_S(\mathbf{l}_i|q_i)}.$$

Proof. See Appendix 4.C. □

The efficiency of PI-MA-IS depends on the similarity between P_S and P_T. It is easy to see that importance weights can become very large when there are large differences between these distributions, leading to high variance. As observed by Chen (2005), this variance can be quantified as the ratio between the variance of outcomes under the source distribution and under the target distribution. We empirically assess the efficiency of the estimator under a wide range of source and target distributions in (§4.4).

Note that PI-MA-IS does not depend on the assignments observed in the original data (cf., Eq. 4.10). This means that it can be applied not just to historical data collected using probabilistic interleaving, but to data collected under any arbitrary distribution, as long as the distribution over result lists is known and non-zero for all lists that are possible under the target distribution. This makes it possible to develop new sampling algorithms that can make interleaved comparisons even more efficient. For example, data could be sampled in a way that allows optimal comparisons of a set of more than two rankers, or with the combined goal of maximizing both the quality of the lists presented to users, and the reusability of the collected data. While doing so is beyond the scope of this thesis, it is an important direction for future research.

4.3 Experiments

Our experiments are designed to assess the efficiency of the existing and proposed interleaved comparison methods. All our experiments rely on the simulation framework detailed in Chapter 3. In comparison to previous work, this setup allows evaluating interleaved comparison methods on a large set of ranker pairs in a controlled experiment. Previous work validated interleaved comparisons on real usage data (Chapelle et al., 2012; Radlinski and Craswell, 2010; Radlinski et al., 2008b), which allowed assessment of these methods in a realistic setting but limited the number of possible ranker comparisons. On the other hand, He et al. (2009) used a small number of hand-constructed test cases for their analysis. Our setup falls in between these as it is more controlled than the former, but has fewer assumptions than the latter.

The following subsections detail the experimental procedures used to simulate interleaved comparisons using live data (§4.3.1) and historical data (§4.3.2). In both settings, we run experiments on the $18,919$ queries of the training set of fold 1 of the MSLR-WEB30k Microsoft learning to rank data set (cf., 3.4), and use the click models for 5-point graded relevance judgments as shown in Table 3.1.

To allow comparisons of many ranker pairs, we generate rankers from the 136 individual features provided with the learning to rank data set. This means that our experiments simulate the task of comparing the effectiveness of individual features for retrieval using varying amounts of historical data, or a combination of historical and live data. As specified in Definition 4.1.4, we compare the efficiency of rankers by comparing the accuracy they obtain after observing a sample of a given size. We measure accuracy after observing m queries as the portion of ranker pairs for which an interleaved comparison method correctly predicts the direction of the difference in NDCG. To compute NDCG difference, we use the manual relevance judgments provided with the learning to rank data set. Then, an interleaved comparison method is deemed more efficient than another if it Pareto dominates it (i.e., its accuracy is at least not significantly lower for all sample sizes, and significantly higher for at least one sample size).

4.3.1 Interleaved Comparisons using Live Data

The main goal of our first experiment is to compare the efficiency of interleaved comparison methods in the live data setting. In this setting, we assume that click data can be collected for any interleaved lists generated by an interleaving algorithm. This means that data is collected directly for the target ranker pair being compared. Our experiments for the live data setting are detailed in Algorithm 8.

Algorithm 8 Experiment 1: Interleaved comparisons using *live* data.

```
1: Input: $interleave(\cdot)$, $compare(\cdot)$, $\mathcal{Q}$, $\mathcal{R}$, $\delta_{NDCG}(\cdot,\cdot)$, $m$, $n$
2: $correct[1..m] = zeros(m)$
3: for $i = 1..n$ do
4:    $\mathbf{O} = []$
5:    $q = random(\mathcal{Q})$
6:    Sample target rankers $(r_1, r_2)$ from $\mathcal{R}$ without replacement
7:    for $(j = 1..m)$ do
8:       $(\mathbf{a}, \mathbf{c}, \mathbf{l}) = interleave(q, r_1, r_2)$
9:       $append(\mathbf{O}, compare(r_1, r_2, \mathbf{a}, \mathbf{c}, \mathbf{l}, q))$
10:      if $sign(\sum \mathbf{O}) = sign(\delta_{NDCG}(r_1, r_2))$ then
11:         $correct[j] + +$
12: return $correct[1..m]/n$
```

The experiment receives as input two functions *interleave* and *compare*, which together specify an interleaving method, such as BI in Algorithm 1 (*interleave* in lines 1–12, *compare* in lines 13–17). It also takes as input a set of queries $\mathcal{Q}$, a set of rankers

$\mathcal{R}$, a method δ_{NDCG} which computes the true NDCG difference between two rankers, the maximum number of impressions per run m, and the number of runs n. The experiment starts by initializing a result vector $correct$ which keeps track of the number of correct decisions of the interleaving method after each impression (line 2). Then, for each run, a query and target ranker pair are sampled from $\mathcal{Q}$ and $\mathcal{R}$ (lines 5–6). The target ranker pair is sampled without replacement, i.e., a ranker cannot be compared to itself (we also exclude cases for which the rankers have the same NDCG, so that there is a preference between rankers in all cases). Then, m impressions are collected by generating interleaved lists (line 8) and comparing the target rankers using the observed data (line 9). Comparison outcomes are aggregated over impressions to determine if a run would identify the preferred ranker correctly (line 10 and 11). Finally, the accuracy after up to m impressions is obtained by dividing $correct$ by the number of runs n. An efficient ranker obtains a high accuracy after observing few impressions. The results of our experiments for the live data setting are reported in §4.4.1.

4.3.2 Interleaved Comparisons using Historical Data

The goal of our second experiment is to assess the efficiency of interleaved comparison method under historical data. This setting assumes that interleaved lists cannot be directly observed for the target rankers being compared. Instead, interleaving data previously collected using a different but known original ranker pair is available. We simulate this setting by generating original ranker pairs, and collecting data for these original ranker pairs, which is then used to estimate comparison outcomes for the target pair. The detailed procedure is shown in Algorithm 9.

Algorithm 9 Experiment 2: Interleaved comparisons using *historical* data.

```
1: **Input**: $interleave(\cdot)$, $compare(\cdot)$, $\mathcal{Q}$, $\mathcal{R}$, $\delta_{NDCG}(\cdot,\cdot)$, $m$, $n$
2: $correct[1..m] = zeros(m)$
3: for $i = 1..n$ do
4:     $\mathbf{O} = []$
5:     $q = random(\mathcal{Q})$
6:     Sample original pair $(r_{o_1}, r_{o_2})$ and target pair $(r_{t_1}, r_{t_2})$ from $\mathcal{R}$ without replacement
7:     for $j = 1..m$ do
8:         $(\mathbf{a}, \mathbf{c}, \mathbf{l}) = interleave(q, r_{o_1}, r_{o_2})$
9:         $\mathbf{O}[i] = compare(r_{t_1}, r_{t_2}, r_{o_1}, r_{o_2}, \mathbf{a}, \mathbf{c}, \mathbf{l}, q)$
10:        if $sign(\sum \mathbf{O}) = sign(\delta_{NDCG}(r_{t_1}, r_{t_2}))$ then
11:            $correct[j]++$
12: return $correct[1..m]/n$
```

The arguments passed to Algorithm 9, as well as its initialization and overall structure, are identical to those for the live data experiments shown in Algorithm 8. The main differences are in lines 6 to 9. In addition to the target ranker pair, an original ranker pair is randomly sampled, again without replacement so that there is no overlap between the rankers used in a given run (line 6). Then, for each impression, the interleaving data is collected for the original ranker pair (line 8). The target rankers are compared using this data (line 9). Experiment outcomes are computed in terms of accuracy for the target rankers as before. An efficient ranker obtains high accuracy with few historical samples. The results of our experiments for the historical data setting are reported in §4.4.2.

4.4 Results and Discussion

In this section we detail our two experiments and present and analyze the obtained results. Our first experiment examines the efficiency of interleaved comparison methods when comparing rankers using live data (§4.4.1). Our second experiment evaluates interleaved comparison methods using historical data (§4.4.2). In addition to presenting our main results, we analyze the interleaved comparison methods' robustness to noise in user feedback and to varying parameter settings.

4.4.1 Interleaved Comparisons using Live Data

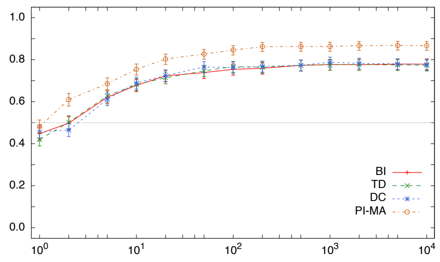
In this section, we present the results of our evaluation of interleaved comparison methods in a live data setting, where interleaving methods interact directly with users. We compare the baseline methods BI, TD, and DC and our proposed method PI-MA, defined as follows:

- **BI**: the balanced interleave method as detailed in Algorithm 1 (§2.3.1), following Chapelle et al. (2012).
- **TD**: the team draft method as detailed in Algorithm 2 (§2.3.1), following Chapelle et al. (2012).
- **DC**: the document constraints method as detailed in Algorithm 3 (§2.3.1), following He et al. (2009).
- **PI-MA**: probabilistic interleaving with marginalization over assignments as defined in Eq. 4.5-4.8 (cf. §4.2.2).

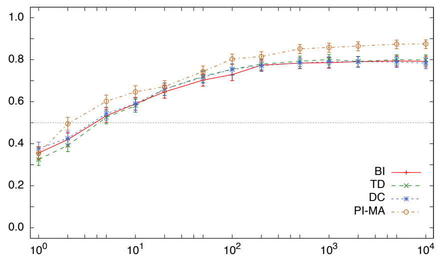
We run experiments for $m = 10{,}000$ impressions, $n = 1{,}000$ times. The experiments use the experimental setup described in §4.3.1.

The results obtained for our four user models are shown in Figure 4.7. Each plot shows the accuracy achieved by each interleaved comparison method over the number of impressions seen for a given user model. The performance of a random baseline would be 0.5, and is marked in grey. Note that the performance of an interleaving method can be below the random baseline in cases where no decision is possible (e.g., the method infers a tie when not enough data has been observed to infer a preference for one of the rankers; the rankers are sampled in such a way that there always is a difference according to the NDCG ground truth). When comparing the efficiency of interleaved comparison methods, we consider both how many impressions are needed before a specific accuracy level is achieved, and what final accuracy is achieved after e.g., 10,000 impressions.

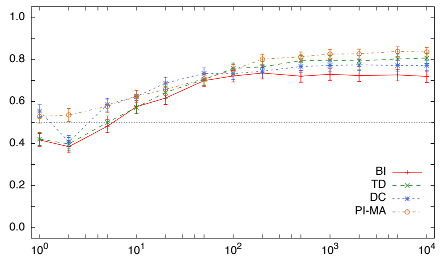
For the *perfect* click model (cf., Figure 4.7(a)) we find that the baseline methods BI, TD and DC achieve close to identical performance throughout the experiment. The final accuracies of these methods after observing 10,000 impressions are 0.78, 0.77, and 0.78



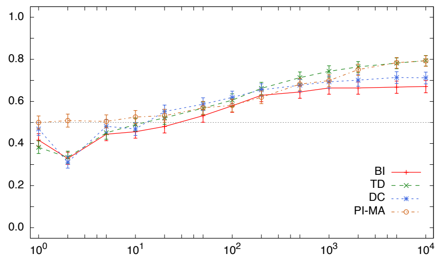
(a) *perfect*



(b) *navigational*



(c) *informational*



(d) *almost random*

Figure 4.7: Results, accuracy of interleaved comparison methods when comparing rankers under live data. Accuracy is computed over 1,000 randomly selected ranker pairs and queries, after 1–10,000 user impressions with varying click models.

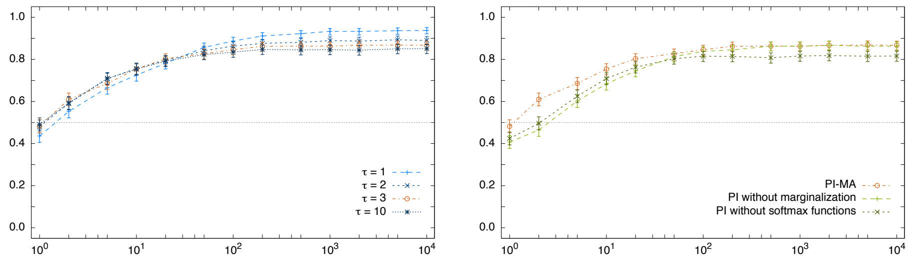
respectively, and there is no significant difference between the methods. We conclude that these methods are similarly efficient when comparing rankers on highly reliable live data. Our proposed method PI-MA outperforms all baseline methods on live data under the *perfect* click model by a large and statistically significant margin. After observing only 50 impressions, PI-MA can more accurately distinguish between rankers than either of the other methods after observing 10,000 impressions. Its final accuracy of 0.87 is significantly higher than that of all baselines. Compared to the best-performing baseline (here, BI), PI-MA can correctly detect a preference on 11.5% more ranker pairs after observing 10,000 impressions.

Results for the *navigational* click model are shown in Figure 4.7(b). In comparison to the *perfect* click model, this model has a higher position bias (higher stop probabilities), and a steeper decay of click probabilities (quadratic, so that the difference between the highest relevance grades is relatively bigger than under the *perfect* click model). The increase in position bias is expected to lead to a decrease in efficiency (this effect was identified for BI, TD, and DC in He et al. (2009)). This effect is confirmed by our results, which can be seen in the slower increase in accuracy as compared to the *perfect* click model. For example, under the *navigational* click model, approximately 50 impressions are needed before all interleaved comparison methods achieve an accuracy of at least 0.7, while for the *perfect* model, only about 20 impressions need to be observed for the same level of accuracy. The steeper decay in click probabilities is expected to lead to click data that better corresponds to the implementation of gain values in NDCG than the linear decay implemented in the *perfect* click model. We find that the accuracy of all methods after 10,000 iterations is slightly higher under the *navigational* model (the accuracy for BI is 0.79, for TD 0.80, for DC 0.78, and for PI-MA 0.88), but none of the differences is statistically significant. We can conclude that under the *navigational* model, interleaving methods have lower efficiency (due to increased position bias), but they converge to at least the same level of accuracy (possibly slightly higher, due to the better match with NDCG gain values) as under the *perfect* click model. Comparing the individual methods, we again find that PI-MA performs significantly better than all baseline methods. The increase in accuracy after 10,000 impressions is 10%.

The *informational* click model has a level of position bias that is similar to that of the *navigational* click model, but a higher level of noise. Thus, users consider more documents per query, but their click behavior makes documents more difficult to distinguish. Figure 4.7(c) shows the results for this click model. The interleaving methods' efficiency is similar to that under the *navigational* model for small sample sizes, with all methods achieving an accuracy of 0.7 within 50 samples. However, the increase in noise affects efficiency for bigger samples. After 10,000 impressions, BI achieves an accuracy of 0.72 (TD - 0.81, DC - 0.77, and PI-MA - 0.84). The performance of BI and of PI-MA is significantly lower than under the *navigational* model. The performance of PI-MA is significantly higher than that of BI and DC under the *informational* model, and higher (but not significantly so) than that of TD. The performance of BI appears to be particularly strongly affected by noise. This method performs significantly worse than all other interleaved comparison methods in this setting. Outcomes computed under this method rely on rank-differences at the lowest-clicked document. As individual clicks become less reliable, so do the comparison outcomes.

Results for the *almost random* click model reflect the performance of interleaved comparison methods under high noise and high position bias (Figure 4.7(d)). As expected, we find that efficiency decreases substantially for all methods. For example, TD is the first method to achieve an accuracy of 0.7 after 500 impressions. After 10,000 impressions, BI achieves an accuracy of only 0.67 and the accuracy of DC is 0.71. TD appears to be the most robust against this form of noise, maintaining an accuracy of 0.79. PI-MA performs better than the baseline methods on small sample sizes, because marginalization helps avoid noisy inferences. Its performance after 10,000 impressions is the same as for TD. In general, PI-MA is expected to converge to the same results as TD in settings with high noise and high position bias, such as the one simulated here. In these settings, the method cannot accurately trade-off between clicks at different positions.

Our results for the different user models indicate that PI-MA Pareto dominates the baseline methods in terms of performance. Under reliable click feedback, the baseline methods perform similarly well, while PI-MA is substantially more efficient at all sample sizes. The reason is that PI-MA can trade off differences between ranks more accurately. For all methods, efficiency decreases as position bias increases, which is in line with earlier work. Increasing noise affects the interleaving methods differently. BI appears to be affected the most strongly, followed by DC. TD is relatively robust to noise. PI-MA reduces to TD under high levels of noise. None of the baseline methods was found to be significantly more accurate than PI-MA at any sample size or level of click noise. We conclude that PI-MA is more efficient than the baseline methods.



(a) PI-MA with varying settings of τ (b) PI without marginalization / softmax functions

Figure 4.8: Results, accuracy of variants of PI-MA in the live data setting and under the *perfect* click model. Accuracy is computed on 1,000 randomly selected ranker pairs and queries, after 1–10,000 user impressions using PI-MA with varying τ, and without softmax functions / marginalization.

After comparing PI-MA to the baseline methods, we now turn to analyzing PI-MA in more detail. PI-MA has one parameter τ. This parameter affects the trade-off between clicked documents at different ranks, similar to the position discount in NDCG. Low values of τ result in slightly more randomization in the constructed interleaved result lists, which means that documents at lower ranks have a higher chance of being placed in the top of the result list and are more likely to be clicked. When comparing interleaving outcomes to NDCG difference, we expect more accurate results for smaller values of τ, as NDCG uses a relatively weak position discount (namely $\log(r)$).

Our analysis is confirmed by our results in Figure 4.8(a) (here: *perfect* click model). For settings of τ that are smaller than the default value $\tau = 3$ (i.e., $\tau \in (1, 2)$, accuracy is higher than for the default settings. Increasing the parameter value to $\tau = 10$ decreases accuracy. While all parameter settings $\tau > 0$ result in an interleaved comparison method that exhibits fidelity as defined in Definition 4.1.2, an appropriate value needs to be chosen when applying this method. Higher values place more emphasis on even small differences between rankings, which may be important in settings where users are typically impatient (e.g., for navigational queries). In settings where users are expected to be more patient, or tend to explore results more broadly, a lower value should be chosen. In comparison, the baseline methods BI, TD, and DC make implicit assumptions about how clicked documents at lower ranks should be weighted, but do not allow the designer of the retrieval system to make this decision explicit.

Finally, we analyze PI-MA in more detail by evaluating its performance after removing individual components of the method. Figure 4.8(b) shows PI-MA ($\tau = 3$), compared to PI-MA without marginalization, and without softmax functions. We find that the complete method has the highest efficiency, as expected. Without marginalization, comparisons are less reliable, leading to lower initial efficiency. The performance difference is compensated for with additional data, confirming that PI and PI-MA converge to the same comparison outcomes. When deterministic ranking functions are used instead of softmax functions, we observe lower efficiency. Without softmax functions, PI-MA does not trade off between differences at different ranks, leading to lower agreement with

NDCG. We conclude that PI-MA is more efficient than variants of the method without marginalization, and without softmax functions. This result confirms the results of our analysis in §4.2.

4.4.2 Interleaved Comparisons using Historical Data

In this section, we evaluate interleaved comparison methods in a historical data setting, where only previously observed interaction data is available. Our experiments do not focus on how to collect such data, but rather assumes that data is available from previous experiments and the task is to use this data effectively. We compare the following methods for interleaved comparisons using historical data:

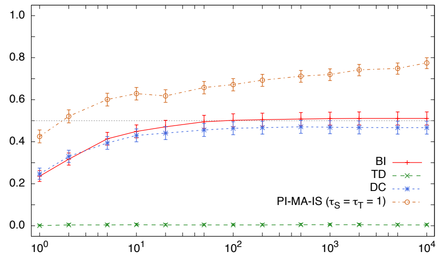
- **BI**: directly applies BI to historical data, as discussed in §4.1.3.
- **TD**: applies TD to all assignments that match historical data, as discussed in §4.1.4.
- **DC**: directly applies DC to historical data, as discussed in §4.1.5.
- **PI-MA-IS**: our full importance sampling estimator with marginalization over assignments, as defined in Eq. 4.10 (cf., §4.2.3). Note that unless specified otherwise, we use a setting of $\tau = 1$ for both the source and the target distribution.

We use the experimental setup described in §3, and the procedure detailed in §4.3.2. Each run is repeated $n = 1{,}000$ times and has a length of $m = 10{,}000$ impressions. Also, for each run, we collect historical data using a randomly selected source ranker pair, and use the collected data to infer information about relative performance of a randomly selected target ranker pair.

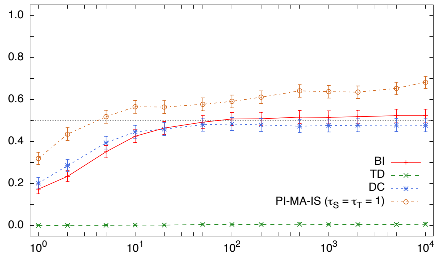
In comparison to the live data setting, we expect interleaved comparison methods to have lower efficiency. This is particularly the case for this setting where source and target distributions can be very different from each other. When source and target distributions are more similar to each other (such as learning to rank settings), efficiency under historical data is expected to be much higher, so the results presented here constitute a lower bound on performance.

Figure 4.9 shows the results obtained in the historical data setting. For the *perfect* click model (Figure 4.9(a)), we see the following performance. BI shows close to random performance, and its performance after 10,000 impressions is not statistically different from the random baseline. DC stays significantly below random performance. These results suggest that the two methods cannot use historical data effectively, even under very reliable feedback. The reason is that differences between the observed interleaved lists and the lists that would be generated by the target rankers cannot be compensated for. TD shows very low accuracy, close to zero. This result confirms our analysis that indicated that this method cannot reuse a large portion of the historical data. Since few lists are useable by this method, most comparisons result in a tie between the compared target rankers.

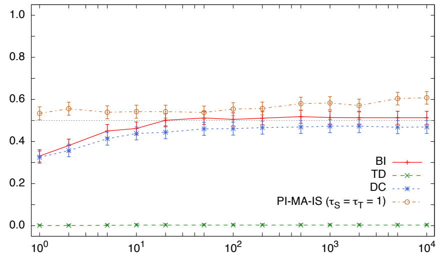
The results in Figure 4.9(a) confirm that PI-MA makes it possible to effectively reuse previously collected data. After 10,000 impressions, this method achieves an accuracy of 0.78. Following the trend of this experiment, accuracy is expected to continue to increase as more impressions are added.



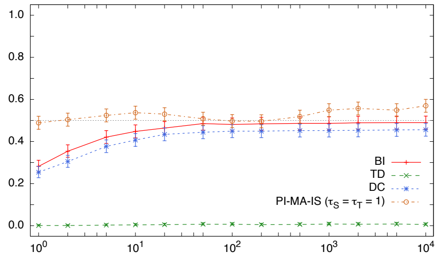
(a) *perfect*



(b) *navigational*



(c) *informational*



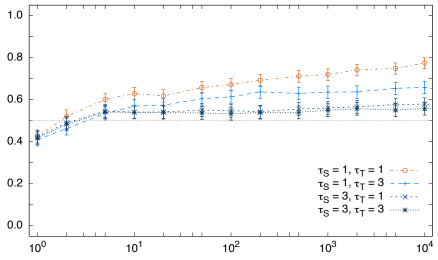
(d) *almost random*

Figure 4.9: Results, accuracy of interleaved comparison methods when comparing rankers under historical data. Accuracy is computed over 1,000 randomly selected ranker pairs and queries, after 1–10,000 user impressions with varying click models.

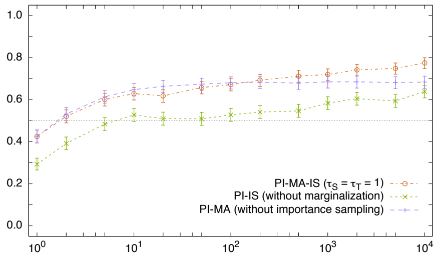
The relative performance of the interleaved comparison methods is the same for all investigated click models. In comparison to the *perfect* click model, the efficiency of PI-MA-IS decreases with increasing click noise as expected. However, the method performs significantly better than the baseline methods under all levels of noise. For the *navigational* model, performance after 10,000 impressions is 0.68 (Figure 4.9(b)), for the *informational* model it is 0.61 (Figure 4.9(c)), and for the *almost random* model 0.57 (Figure 4.9(d)). This shows that efficiency degrades gracefully with increases in noise. For high levels of noise (such as under the *almost random* click model) the required amount of data can be several orders of magnitude higher than under the *perfect* click model to obtain the same level of accuracy. Performance of the baseline methods in the historical data setting does not appear to be substantially affected by noise.

After comparing interleaved comparison methods in the historical feedback setting, we turn to analyzing the characteristics of PI-IS-MA in more detail. First, we investigate the effect of choosing different values of τ during data collection and inference (Figure 4.10(a)).

Under historical data, τ has several effects. For the source rankers (τ_S), it determines the level of exploration during data collection. As $\tau_S \rightarrow \infty$, the level of exploration approaches random exploration. A high level of exploration ensures that result lists that are likely under the target rankers are sufficiently well covered during data collection, which reduces variance in the later comparison stage. This is confirmed by comparing



(a) PI-MA-IS with varying settings of τ



(b) PI-MA-IS vs. PI without marginalization / importance sampling

Figure 4.10: Results, accuracy of variants of PI-MA-IS in the historical data setting and under the *perfect* click model. Accuracy is computed on 1,000 randomly selected ranker pairs and queries, after 1–10,000 user impressions using PI-MA-IS (a) with varying τ_S and τ_T, and (b) compared to PI-IS (without marginalization) and PI-MA (without importance sampling).

our results for PI-MA-IS with the parameter setting $\tau_S = 1, \tau_T = 3$ to those for the setting $\tau_S = 3, \tau_T = 3$. Data collection in the first setting is more exploratory, which leads to a significant increase in efficiency.

Changing τ for the target distribution (τ_T) also has an effect on variance, although it is weaker than that observed for the source distribution. Two factors play a role here. First, smaller values of τ_T lead to comparisons that more accurately correspond to NDCG position discounts (cf., §4.4.1, Figure 4.8(a)). Second, smaller values of τ_T make the target distribution slightly broader, resulting in smaller differences between the source and target distributions and therefore smaller importance weights. The relative importance of these two effects can be estimated with the help of our results obtained in the live setting. There, the accuracy for $\tau = 1$ after 10,000 impressions is substantially (7.5%) and significantly higher than for $\tau = 3$. Under historical data, performance for the setting $\tau_S = 1, \tau_T = 1$ is also significantly higher than for the setting $\tau_S = 1, \tau_T = 3$. Here, the increase is 17.6%, more than twice as high as in the live setting. We conclude that a large portion of this increase is due to the reduced distance between source and target distribution and the resulting reduction in variance. Finally, when comparing settings with low exploration under the source distribution ($\tau_S = 3$), performance differs only marginally. This suggests that a high amount of exploration during data collection is crucial for achieving high efficiency of PI-IS-MA.

Finally, we examine how different components of PI-IS-MA contribute to the performance of this method under historical data. Figure 4.10(b) shows our previous results for PI-IS-MA and for the following additional runs:

- **PI-IS**: PI that uses the naive importance sampling estimator in Eq. 4.9 to compensate for differences between source and target distribution (cf., §4.2.3).
- **PI-MA**: directly applies PI-MA as defined in Eq. 4.5-4.8 (cf. §4.2.2), without compensating for differences between source and target distributions.

Our results confirm the outcomes of our analysis and derivation of PI-MA-IS (cf., 4.2.3). The variant PI-IS (i.e., without marginalization) is significantly less efficient than the full method PI-IS-MA. This confirms that marginalization is an effective way to compensate for noise. The effect is much stronger than in the live data setting because, under historical data, the level of noise is much higher (due to the variance introduced by importance sampling). In the limit, we expect that the performance of PI-IS converges to the same value as PI-IS-MA, but after 10,000 impressions its accuracy is 0.639, 17.5% lower. If PI-MA is applied without importance sampling, it performs as well as PI-IS-MA for small sample sizes. However, we also observe the bias introduced under this method, as it converges to a lower accuracy after processing approximately 200 impressions, making it less efficient in the long run. Performance of PI-MA when applied to historical data is found to be 0.68 after 10,000 impressions, 12% lower than that of PI-MA-IS. These results demonstrate that PI-MA-IS successfully compensates for bias while maintaining high efficiency.

To summarize, our experiments in the historical data setting confirm that PI-MA-IS can effectively reuse historical data for inferring interleaved comparison outcomes. Alternatives based on existing interleaved comparison methods are not able to do this effectively, due to data sparsity and bias. The efficiency of PI-MA-IS under historical data is found to decrease as click noise increases, as expected. More detailed analysis shows that choosing a sufficiently exploratory source distribution is crucial for obtaining good performance. Finally, our results show that marginalization and importance sampling contribute to the effectiveness of PI-MA-IS as suggested by our analysis.

4.5 Conclusion

In this chapter, we introduced a framework for analyzing interleaved comparisons methods, analyzed existing methods, and proposed a probabilistic interleaved comparison method that addresses some of the challenges raised in our analysis. The proposed framework characterizes interleaved comparison methods in terms of fidelity, soundness, and efficiency. Fidelity reflects whether a method measures what it is intended to measure, soundness refers to its statistical properties, and efficiency reflects how much sample data a method requires to make comparisons. This framework is a step towards formalizing the requirements for interleaved comparison methods. It allows us to make more concrete statements about how interleaved comparison methods should behave than previously possible.

We analyzed existing interleaved comparison methods using the proposed framework, and found that none exhibit a minimal requirement of fidelity, namely that the method prefers rankers that rank clicked documents higher. We then proposed a new interleaved comparison method, probabilistic interleave, and showed that it does exhibit fidelity. Next, we devised several estimators for probabilistic interleave, and proved their statistical soundness. These estimators included a naive estimator (PI), a marginalized estimator designed to improve efficiency by reducing variance (PI-MA), and an estimator based on marginalization and importance sampling (PI-MA-IS) that makes it possible to reuse previously collected (historical) data.

We empirically confirmed the results of our analysis through a series of experiments that simulate user interactions with a retrieval system using a fully annotated learning to rank data set and click models. Our experiments in the live data setting showed that PI-MA is more efficient than all existing interleaved comparison methods. Further, experiments on different variants of PI-MA confirmed that PI-MA with marginalization and softmax functions is more efficient than variants without either component. In our experiments with simulated historical click data, we found that PI-MA-IS can effectively reuse historical data. Due to the increase in noise due to importance sampling, efficiency is lower than under live data, as expected. We also experimentally confirmed that the difference between the source and target distributions has a strong effect on the efficiency of PI-MA-IS.

This chapter focused primarily on interleaved comparison methods' theoretical properties and on investigating their effectiveness in a controlled experimental setup. Our analysis and experiments explicitly made a number of assumptions about the relationship between relevance and user click behavior. These assumptions were based on earlier work on click models, but there is still a large gap between the current models and the very noisy observations of user behavior in real (web) search environments. As a step towards testing these assumptions, we investigate interleaved comparison methods in a real-world (web) search setting in the next chapter (Chapter 5). In particular, we investigate whether and how search result presentation affects users' click behavior (caption bias), and how these effects influence interleaved comparison outcomes.

We further follow-up on the work presented in this chapter by integrating our method for estimating interleaved comparison outcomes from historical data, PI-MA-IS, with an online learning to rank approach (in Chapter 7). In the present chapter, PI-MA-IS was assessed theoretically, and in an online evaluation setting where source and target distribution could be very different from another, and we were able to show that the method can effectively reuse historical data. We expect higher efficiency in online learning to rank settings, where the differences between source and target rankers are typically small, resulting in low variance of the estimated comparison outcomes. This hypothesis is tested in Chapter 7, where we devise the first two methods for learning with historical data reuse, based on PI-MA and PI-MA-IS.

4.A Proof of Theorem 4.1.1

Theorem 4.1.1. *The estimator in Eq. 4.2 is equal to two times the sample mean (Eq. 4.1).*

Proof. Below, we use the fact that $\frac{1}{n}\sum_{i=0}^{n} o_i = \frac{1}{n} wins(\mathbf{l}_2) - wins(\mathbf{l}_1)$ (following from the definition of $wins(\mathbf{l}_i)$ ($o = -1$ and $o = +1$ for $\mathbf{l}_1$ and $\mathbf{l}_2$ respectively) and $ties(\mathbf{l}_{1,2})$ ($o = 0$) (cf., Chapter 2), and that the number of samples is $n = wins(\mathbf{l}_1) + wins(\mathbf{l}_2) + ties(\mathbf{l}_{1,2})$.

$$\begin{aligned}
2\hat{E}_{wins} &= 2\left(\frac{wins(\mathbf{l}_2) + \frac{1}{2}ties(\mathbf{l}_{1,2})}{wins(\mathbf{l}_2) + wins(\mathbf{l}_1) + ties(\mathbf{l}_{1,2})} - 0.5\right) \\
&= 2\left(\frac{wins(\mathbf{l}_2) + \frac{1}{2}ties(\mathbf{l}_{1,2})}{n} - \frac{\frac{1}{2}n}{n}\right) \\
&= \frac{1}{n}\left(2 \sim wins(\mathbf{l}_2) + ties(\mathbf{l}_{1,2}) - (wins(\mathbf{l}_2) + wins(\mathbf{l}_1) + ties(\mathbf{l}_{1,2}))\right) \\
&= \frac{1}{n}\left(wins(\mathbf{l}_2) - wins(\mathbf{l}_1)\right) = \frac{1}{n}\sum_{i=0}^{n} o_i.
\end{aligned}$$

□

4.B Proof of Theorem 4.2.1

Theorem 4.2.1. *The following estimator is unbiased and consistent given samples from an interleaving experiment conducted according to the graphical model in Figure 4.1(b) (Eq. 4.5):*

$$\hat{E}[O] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{a}\in\mathbf{A}}\sum_{o\in O} oP(o|\mathbf{a},\mathbf{c}_i)P(\mathbf{a}|\mathbf{l}_i, q_i).$$

Proof. We start by defining a new function f:

$$f(\mathbf{C},\mathbf{L},Q) = \sum_{\mathbf{a}\in\mathbf{A}}\sum_{o\in O} oP(o|\mathbf{C},\mathbf{a})P(\mathbf{a}|\mathbf{L},Q).$$

Note that Eq. 4.5 is just the sample mean of $f(\mathbf{C},\mathbf{L},Q)$ and is thus an unbiased and consistent estimator of $E[f(\mathbf{C},\mathbf{L},Q)]$. Therefore, if we can show that $E[O] = E[f(\mathbf{C},\mathbf{L},Q)]$, that will imply that Eq. 4.5 is also an unbiased and consistent estimator of $E[O]$.

We start with the definition of $E[O]$:

$$E[O] = \sum_{o\in O} oP(o).$$

$P(O)$ can be obtained by marginalizing out the other variables:

$$P(O) = \sum_{\mathbf{a}\in\mathbf{A}}\sum_{\mathbf{c}\in\mathbf{C}}\sum_{\mathbf{l}\in\mathbf{L}}\sum_{q\in Q} P(\mathbf{a},\mathbf{c},\mathbf{l},q,O),$$

where, according to the graphical model in Figure 4.1(b), $P(\mathbf{A}, \mathbf{C}, \mathbf{L}, Q, O) = P(O|\mathbf{C}, \mathbf{A})$ $P(\mathbf{C}|\mathbf{L}, Q)$ $P(\mathbf{L}|\mathbf{A}, Q)$ $P(\mathbf{A})P(Q)$. Thus, we can rewrite $E[O]$ as

$$E[O] = \sum_{\mathbf{a}\in\mathbf{A}} \sum_{\mathbf{c}\in\mathbf{C}} \sum_{\mathbf{l}\in\mathbf{L}} \sum_{q\in Q} \sum_{o\in O} oP(o|\mathbf{a}, \mathbf{c})P(\mathbf{c}|\mathbf{l}, q)P(\mathbf{l}|\mathbf{a}, q)P(\mathbf{a})P(q).$$

Observing that $P(\mathbf{L}|\mathbf{A}, Q) = \frac{P(\mathbf{A}|\mathbf{L},Q)P(\mathbf{L}|Q)}{P(\mathbf{A}|Q)}$ (Bayes rule) and $P(\mathbf{A}|Q) = P(\mathbf{A})$ ($\mathbf{A}$ and Q are independent) gives us

$$E[O] = \sum_{\mathbf{a}\in\mathbf{A}} \sum_{\mathbf{c}\in\mathbf{C}} \sum_{\mathbf{l}\in\mathbf{L}} \sum_{q\in Q} \sum_{o\in O} oP(o|\mathbf{a}, \mathbf{c})P(\mathbf{a}|\mathbf{l}, q)P(\mathbf{c}|\mathbf{l}, q)p(\mathbf{l}|q)P(q).$$

Figure 4.1(b) implies $P(\mathbf{C}, \mathbf{L}, Q) = P(\mathbf{C}|\mathbf{L}, Q)P(\mathbf{L}|Q)P(Q)$, yielding:

$$E[O] = \sum_{\mathbf{a}\in\mathbf{A}} \sum_{\mathbf{c}\in\mathbf{C}} \sum_{\mathbf{l}\in\mathbf{L}} \sum_{q\in Q} \sum_{o\in O} oP(o|\mathbf{a}, \mathbf{c})P(\mathbf{a}|\mathbf{l}, q)P(\mathbf{c}, \mathbf{l}, q).$$

From the definition of $f(\mathbf{C}, \mathbf{L}, Q)$ this gives us:

$$E[O] = \sum_{\mathbf{c}\in\mathbf{C}} \sum_{\mathbf{l}\in\mathbf{L}} \sum_{q\in Q} f(\mathbf{c}, \mathbf{l}, q)P(\mathbf{c}, \mathbf{l}, q),$$

which is the definition of $E[f(\mathbf{C}, \mathbf{L}, Q)]$, so that:

$$E[O] = E[f(\mathbf{C}, \mathbf{L}, Q)].$$

□

4.C Proof of Theorem 4.2.2

Theorem 4.2.2. *The following estimator is unbiased and consistent given samples from an interleaving experiment conducted according to the graphical model in Figure 4.1(b) under P_S:*

$$\hat{E}_T[O] = \frac{1}{n} \sum_{i=1}^{n} \sum_{\mathbf{a}\in\mathbf{A}} \sum_{o\in O} oP_T(o|\mathbf{a}, \mathbf{c}_i)P_T(\mathbf{a}|\mathbf{l}_i, q_i)\frac{P_T(\mathbf{l}_i|q_i)}{P_S(\mathbf{l}_i|q_i)}.$$

Proof. As in Theorem 4.2.1, we start by defining f:

$$f(C, L, Q) = \sum_{\mathbf{a}\in\mathbf{A}} \sum_{o\in O} oP_T(o|\mathbf{a}, \mathbf{C})P_T(\mathbf{a}|\mathbf{L}, Q).$$

Plugging this into the importance sampling estimator in Eq. 2.3 gives:

$$\hat{E}_T[O] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{c}_i, \mathbf{l}_i, q_i)\frac{P_T(\mathbf{c}_i, \mathbf{l}_i, q_i)}{P_S(\mathbf{c}_i, \mathbf{l}_i, q_i)},$$

which is unbiased and consistent if $P_S(\mathbf{C}, \mathbf{L}, Q)$ is non-zero at all points at which $P_T(\mathbf{C}, \mathbf{L}, Q)$ is non-zero. Figure 4.1(b) implies that $P(\mathbf{C}, \mathbf{L}, Q) = P(\mathbf{C}|\mathbf{L}, Q)P(\mathbf{L}|Q)P(Q)$, yielding:

$$\hat{E}_T[O] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{c}_i, \mathbf{l}_i, q_i)\frac{P_T(\mathbf{c}_i|\mathbf{l}_i, q_i)P_T(\mathbf{l}_i|q_i)P_T(q_i)}{P_S(\mathbf{c}_i|\mathbf{l}_i, q_i)P_S(\mathbf{l}_i|q_i)P_S(q_i)}.$$

Because we assume that clicks and queries are drawn from the same static distribution, independent of the ranker pair used to generate the presented list, we know that $P_T(Q) = P_S(Q)$ and $P_T(\mathbf{C}|\mathbf{L}, Q) = P_S(\mathbf{C}|\mathbf{L}, Q)$, giving us:

$$\hat{E}_T[O] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{c}_i, \mathbf{l}_i, q_i)\frac{P_T(\mathbf{l}_i|q_i)}{P_S(\mathbf{l}_i|q_i)}.$$

From the definition of $f(\mathbf{C}, \mathbf{L}, Q)$ we obtain:

$$\hat{E}_T[O] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{a}\in\mathbf{A}}\sum_{o\in O} oP_T(o|\mathbf{a}, \mathbf{c}_i)P_T(\mathbf{a}|\mathbf{l}_i, q_i)\frac{P_T(\mathbf{l}_i|q_i)}{P_S(\mathbf{l}_i|q_i)}.$$

To show that $P_S(\mathbf{C}, \mathbf{L}, Q)$ is non-zero whenever $P_T(\mathbf{C}, \mathbf{L}, Q)$ is non-zero, we need only show that $P_S(\mathbf{L}|Q)$ is non-zero at all points at which $P_T(\mathbf{L}|Q)$ is non-zero. This follows from three facts already mentioned above: 1) $P(\mathbf{C}, \mathbf{L}, Q) = P(\mathbf{C}|\mathbf{L}, Q)P(\mathbf{L}|Q)P(Q)$, 2) $P_T(Q) = P_S(Q)$, and 3) $P_T(\mathbf{C}|\mathbf{L}, Q) = P_S(\mathbf{C}|\mathbf{L}, Q)$. Figure 4.1(b) implies that $P(\mathbf{L}|Q) = \sum_{\mathbf{a}\in\mathbf{A}} P(\mathbf{L}|\mathbf{a}, Q)$ (Eq. 4.8), which is non-zero if $P(\mathbf{L}|\mathbf{A}, Q)$ is non-zero for at least one assignment. From the definition of the interleaving process (Eq. 4.7) we have that $P_S(\mathbf{L}|\mathbf{A}, Q)$ is non-zero for all assignments. $\square$

5 Caption Bias in Interleaving Experiments

In the previous chapter we focused on the theoretical properties of interleaved comparison methods. Here, we focus on applying these methods to obtain feedback in a real-live setting, web search. When we apply interleaved comparison methods for online evaluation, or online learning to rank in a search setting, we typically expect to obtain an estimate of the relative quality of rankers in terms of how they rank relevant results. Interleaved comparison methods have been developed to obtain such estimates in the face of position bias. Beyond compensating for position bias, these methods assume that user clicks reflect relevance, although this relation may be noisy.

While interleaved comparison methods promise to reflect true user preferences, their reliance on user clicks makes them susceptible to click bias[1] (cf., §2.3.2) when the assumptions that these methods are based on are violated. For example, users have been previously shown to be more likely to click on results with attractive titles and snippets (Clarke et al., 2007; Yue et al., 2010b). An interleaved comparison where one ranker tends to generate results that attract more clicks (without being more relevant) may thus detect a preference for the wrong ranker.

This is the problem that we address in this chapter: *How are interleaving outcomes affected by differences in result presentation in practice?* On the one hand, as previous work has assumed, caption bias may affect rankers equally. This would increase variance when computing interleaved comparison outcomes but not introduce bias. On the other hand, typical ranker optimization changes may affect captions (for example, by favoring titles with more highlighting), thereby creating a systematic effect on click behavior. When interleaving methods are applied to measure preferences between rankers, it is important to identify when caption bias may be occurring, and to be able to avoid it or compensate for it. Specifically, we address the following three research questions:

RQ 8 (How) does result presentation affect user clicks (caption bias)?

RQ 9 Can we model caption bias, and compensate for it in interleaving experiments?

RQ 10 (How) does caption bias affect interleaving experiments?

[1]We use *click bias* to refer to any characteristic of a search result that systematically influences click behavior in such a way that a result receives more or fewer clicks than would be warranted by the item's content-based relevance to the query alone. We use *caption bias* to refer to forms of click bias related to the visual presentation of results on a search result page.

We address these questions as follows. First, we introduce a general probabilistic, feature-based approach for modeling caption bias in user clicks. We propose two types of features to instantiate this model based on (1) an assumption that caption bias independently affects clicks on each document, and (2) modeling interactions between caption effects on nearby documents. Next, we show that the developed caption-bias models can be integrated with existing interleaved comparison methods, by devising alternative estimators for TD and PI-MA. Finally, we apply this approach to real interleaving experiments, finding that the method identifies caption bias when expected and produces de-biased interleaving outcomes.

The results of our analysis have implications for how and in what cases interleaving methods can be applied in practice. In particular, the work presented in this chapter contributes to a better understanding of IR evaluation using interleaving methods, and to making them more reliable and robust.

The remainder of this chapter is organized as follows. We detail our approach for modeling and compensating for caption bias in §5.1. Our experiments give insights into the types of features that are most effective for modeling caption bias, the effectiveness of our models for predicting click behavior and user preferences, and the effect of caption bias on interleaved comparisons. They are presented and discussed in §5.2. We conclude in §5.3.

5.1 Method

Our method is based on the following idea: when assigning credit for clicks to rankers in an interleaving experiment, the credit can be reweighted to reflect the likelihood of the user clicking on the result based on just caption bias. This is similar to the ideas by (Chapelle et al., 2012; Yue et al., 2010b), although here we focus on improving the fidelity rather than the sensitivity of interleaving experiments.

We reweight clicks by the inverse of their caption-based click odds. This means that results that are very likely to be clicked due simply to their visual characteristics receive a low weight, while higher weights are assigned to results whose representation is less likely to attract clicks. Thus, clicks on relatively less “clickable” results are taken to provide a more reliable indication of relevance, while more “clickable” results are considered prone to attracting clicks unwarranted by their relevance and receive a lower weight. This principle is implemented in the following 3-step approach: (1) model the probability of a click as a combination of position, relevance and caption bias, (2) learn the weights of this model using observations of past user behavior, and (3) factor out the caption bias component from interleaving evaluations to make clicks better reflect relevance. Below, we detail our caption bias model (§5.1.1) and features (§5.1.2), as well as our approach for reweighting clicks in interleaved comparison methods (§5.1.3).

5.1.1 Modeling Caption Bias

Goal of our model is to relate a set of observations $\mathbf{x}$ (here: features that encode characteristics of a document in a result list) to the probability of that document being clicked by a user. A natural model of such a relation, with minimal assumptions, is the logistic regression model (Friedman et al., 2001). Such a model ensures that predictions are in the form of a probability distribution, and it allows a straightforward interpretation of trained regression weights in terms of their effect on the likelihood of an outcome. Because our hypothesis is that a model that includes caption bias features can more accurately predict click probabilities than one with only relevance and position features, we explicitly formulate our model in terms of relevance features $\mathbf{x}_r$, position features $\mathbf{x}_p$, and caption features $\mathbf{x}_c$. This results in the following model:

$$P(C|\mathbf{x}_r, \mathbf{x}_p, \mathbf{x}_c) = \frac{1}{1 + e^{-\beta_0 - \beta_r \mathbf{x}_r - \beta_p \mathbf{x}_p - \beta_c \mathbf{x}_c}}. \tag{5.1}$$

Here, $P(C|\mathbf{x}_r, \mathbf{x}_p, \mathbf{x}_c)$ denotes the probability of a click on a result document, that is characterized by relevance features $\mathbf{x}_r$, position features $\mathbf{x}_p$, and caption features $\mathbf{x}_c$. The parameters of the model — the intercept β_0, and the coefficients β_r, β_p, and β_c — are estimated from training data using maximum likelihood estimation. While a model that takes into account nonlinear combinations of bias features may produce more accurate results, this model is easy to interpret, less prone to overfitting than more complex models, and we find it to perform well when validated on the task of predicting clicks (cf., §5.2.2).

The weights obtained after training the model in Eq. 5.1 can be interpreted in terms of the effect of the corresponding feature on the click odds, a characteristic we make use of when applying the trained model to reweight clicks as shown in §5.1.3. Note that only caption features $\mathbf{x}_c$ are used for reweighting, to compensate for caption bias. The remaining features are included during model training only, to remove effects of document relevance and position. In this way we obtain a model of a document's click likelihood given its presentation. The caption features used in this study are detailed in §5.1.2. Relevance and position features are described below.

The relevance level of a document d to a query q is modeled by $\mathbf{x}_r = \phi_r(d|q)$ as a vector of five binary features. They represent 5-point relevance judgments that range from "not relevant" to "highly relevant". Our position features $\mathbf{x}_p = \phi_p(d|q)$ follow the formulation in (Yue et al., 2010b). Specifically, we use six binary indicator features that indicate whether each document was presented at rank 1, 2, 3, 4 to 5, 6 to 9, or 10 and below.

In a preliminary study, we also considered two alternatives to the model described above. First, we assessed document-wise models that do not take into account relevance information, but simply model caption bias using visual and position features. However, we found that models that do take relevance into account model click behavior more accurately. Second, we evaluated a pairwise model that predicted which of two documents was more likely to be clicked, based on features that captured visual differences between them (i.e., predicting which of two documents is more likely to be clicked, similarly to the approaches by Clarke et al. (2007) and Yue et al. (2010b)). Here, we found the per-document formulation in Eq. 5.1 to be much more effective in explaining click behavior. Therefore, we focus on this model in the rest of this chapter. Nevertheless, we found that pairwise features, that capture the relationship between the document for which clicks are predicted and its neighboring documents, can be combined with our per-document model to further improve performance (implemented as pairwise features, cf., §5.1.2).

5.1.2 Caption Bias Features

We use two types of visual features to model caption bias: per-document features and pairwise features (both are encoded in the caption feature vector $\mathbf{x}_c = \phi_c(d|q)$). Both types of features are detailed below. For all features, we assume a standard result page of a web search engine, where results are displayed with their title, URL, and a snippet that shows how each document relates to the user's query.

Per-Document Caption Features

Our per-document features are designed to capture characteristics of individual search result captions, and model aspects that may make them likely to attract (or discourage) clicks. We started with the features investigated in Clarke et al. (2007), such as *short snippet*, *term matches in the title*, and *URL length*.

From the initial set of features, we restricted our features to those that we believe may capture visual characteristics relevant to our task, yet are not likely to be strongly affected by document relevance. In an initial study, we found a statistically significant effect of e.g., the number of query term matches with the document title, and the number of phrase matches with the snippet on click behavior. However, we think that these observations were strongly affected by the rankers used to collect our training data. E.g., a ranker may over- or under-emphasize the importance of matches in the document title, while the 5-point relevance judgments (cf., §5.1.1) used to remove major effects of relevance may not be sufficiently fine-grained to compensate for these ranker effects. To avoid contamination of our caption bias model with such ranker effects, we removed features for which these were a concern.

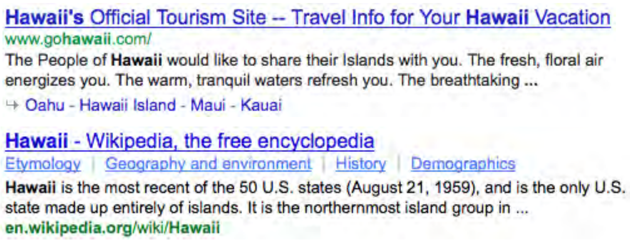
We binarized all per-document features to avoid cases where our caption bias model would be dominated by individual unbounded values. For each "raw" feature (e.g., title length), we started with natural thresholds, such as the first and third quartile, the mean, and points identified by visual inspection of the feature's histogram. Next, binary features representing these bins were added to a model of document relevance and position, which was then trained using logistic regression. The thresholds were then manually tuned to maximize the model's fit to the training data (i.e., thresholds were increased and decreased and the model re-trained, until the magnitude of the residuals from the fitted model did not decrease further).

Finally, all constructed binary features were combined in one model, and features that did not have a significant effect on the models' prediction (measured using a χ^2 test, and $p < 0.001$) were removed from the model. In this step we reduced the number of features from 25 to the final set of 10 per-document features.

Our per-document features are presented in Table 5.1. The feature *deep links* refers to links to subsections of a website that are grouped under a main title result as illustrated in Figure 5.1. We included this feature because a strong relation with click behavior was found, and this type of presentation is common to most major web search engines. The length-related features *short URL*, *short title*, *long title*, *short snippet*, and *long snippet* were converted to binary values as described above. For longer URLs, the number of slashes was found more informative than the number of characters. Similarly, for title length, the number of words was more informative than the number of characters. For

Feature	Description
Deep links	The result is presented with deep links
Short URL	The length of the displayed URL is 30 or less characters
URL slashes	The URL of the presented result contains more than 5 slashes
URL bold	The presented URL has more than 1 highlighted section
Short title	The presented title consists of less than 3 words
Long title	The presented title consists of more than 7 words
Title start	The title begins with an exact match of the query
Title bold	The title contains more than 2 highlighted sections
Short snippet	The displayed snippet is shorter than 40 characters
Long snippet	The displayed snippet is longer than 170 characters

Table 5.1: Per-document features for capturing visual characteristics of individual result captions.


Hawaii's Official Tourism Site -- Travel Info for Your **Hawaii** Vacation
www.go**hawaii**.com/
The People of **Hawaii** would like to share their Islands with you. The fresh, floral air energizes you. The warm, tranquil waters refresh you. The breathtaking ...
↪ Oahu - Hawaii Island - Maui - Kauai

Hawaii - Wikipedia, the free encyclopedia
Etymology | Geography and environment | History | Demographics
Hawaii is the most recent of the 50 U.S. states (August 21, 1959), and is the only U.S. state made up entirely of islands. It is the northernmost island group in ...
en.wikipedia.org/wiki/**Hawaii**

Figure 5.1: Example search results of two commercial web search engines, with deep links included in addition to the title link.

snippet length, the threshold for *short snippets* corresponds to roughly half a line of text, while the threshold for *long snippets* corresponds to a length where the text would flow onto a third line.

Here, we list only the final features and exclude features where no significant effect on click behavior was detected (e.g., highlighting in the snippet). In addition to those listed, we initially tested the following features proposed in earlier studies (Clarke et al., 2007): the number of query term matches with the title, snippet, and URL respectively, and the respective number of phrase matches. However, because our models were developed to specifically capture changes in click behavior due to visual characteristics, we removed these features that may be more strongly affected by document content.

Pairwise Caption Features

Our second set of features considers not individual documents, but pairs of documents. The intuition behind these features is that documents presented in response to a query

Feature	**Description**
Δ URL length above / below	The difference between the length of the URL of the current document and of the document ranked immediately above it / immediately below it
Δ URL slashes above / below	The difference in the number of slashes in the URL
Δ URL bold above / below	The difference in the number of words highlighted in the URL
Δ title length above / below	The difference in title length (in characters or words)
Δ title bold above / below	The difference in the number of highlighted words or sections in the title
Δ snippet length above / below	The difference in the length of the snippet (in characters or words)
Δ snippet bold above / below	The difference in the number of highlighted words or sections in the snippet

Table 5.2: Pairwise visual features for capturing caption bias.

attract clicks not only based on their own representations, but also depending on other, surrounding documents. For example, a somewhat attractive result may attract clicks when placed next to a poorly presented result, but may not get much attention when placed next to a result with a better presentation. Although we found per-document models to perform better individually, we hypothesized that click behavior could best be captured by a combination of characteristics of a document's own representation, and those of surrounding documents.

As for our per-document features, we avoided unbounded features to prevent individual features from dominating the model (this may happen when e.g., directly including the difference in URL length). We achieved this by encoding all pairwise features as ternary values (i.e., with possible values $(-1, 0, 1)$. Thus, e.g., the feature Δ *URL length above* would be -1 if the URL length of the current document is less than that of the document ranked immediately above it, 0 if there was no difference, and 1 if the URL was longer than that of the document ranked above it.

A complete list of the investigated pairwise features is provided in Table 5.2. The features Δ *title bold above / below (in words)* and Δ *snippet bold above / below (in words)* are designed to capture relationships between neighboring documents that are as close as possible to those explored in (Yue et al., 2010b), so that effects found here can be compared to this earlier work. In addition, we include features that capture differences in highlighting of the URL, and consider the number of highlighted sections (e.g., phrases) in addition to that of individual words. Finally, we add features that capture the length differences of URL, title, and snippet. As for the document-wise features, we exclude features based on term matches, to focus on visual aspects of the search result captions.

5.1.3 Reweighting Clicks

In this section we detail how we apply the caption bias models developed in §5.1.1 to interleaved comparison methods to compensate for caption bias. The key idea is to reweight observed clicks on result documents by the change in click odds for that document that is due to caption bias. We base our method on two interleaved comparison methods, TD (Radlinski et al., 2008b) (cf., §2.3.1) and PI-MA (Chapter 4). Extensions to other interleaved comparison methods are straightforward following the same procedure.

We start from the interleaved comparison outcomes TD and PI-MA, and their estimators of comparison outcomes (the sample mean as defined in Eq. 4.1 for TD, and our estimator in Eq. 4.5 for PI-MA). To make explicit how comparison outcomes are computed given an observed interleaved result list $\mathbf{l}$, clicks $\mathbf{c}$, and assignments $\mathbf{a}$ by these estimators we rewrite comparison outcomes o as[2]

$$o = \sum_{r=1}^{len(\mathbf{l})} \mathbf{c}[r]\mathbf{a}[r] - \sum_{r=1}^{len(\mathbf{l})} \mathbf{c}[r]\bar{\mathbf{a}}[r], \tag{5.2}$$

where $len(\mathbf{l})$ is the length of $\mathbf{l}$, r is the rank of a document in $\mathbf{l}$, and $\mathbf{c}[r] \in \{0,1\}$ indicates whether the document $\mathbf{l}[r]$ was clicked. For brevity of notation, we take $\mathbf{a}[r] \in \{0,1\}$ to indicate whether $\mathbf{l}[r]$ was contributed by ranker $\mathbf{l}_1$, with its complement $\bar{\mathbf{a}}[r] \in \{0,1\}$ indicating whether it was contributed by ranker $\mathbf{l}_2$. Thus $\mathbf{c}[r]\mathbf{a}[r]$ evaluates to 1 iff $\mathbf{l}[r]$ was clicked, and contributed by ranker $\mathbf{l}_1$, and to compute o, we simply take the difference in the number of clicks that were contributed by $\mathbf{l}_1$ and $\mathbf{l}_2$.

Based on the formulation of o in Eq. 5.2 we correct for caption bias using the coefficients β_c obtained from the trained model in Eq. 5.1. The exponential of the coefficient for a feature can be interpreted as the change in the click odds of a document, given that the feature is present (for binary features; coefficients for real-valued features are interpreted as the change in click odds ratio for one unit change). It is an approximation of the change in click probability attributed to that feature.[3] We exploit this relationship by reweighting each observed click on a document by the effect of the documents' caption features on its click odds, $e^{\beta_c \cdot \mathbf{x}_c}$, where $\mathbf{x}_c = \phi_c(d,q)$ is the caption feature vector for the clicked document.

For an example of how click reweighting is applied, consider a result with a very short URL (i.e., *short URL* is true), and 3 highlighted sections in the displayed result title (*title bold*). Also, assume that the estimated weights for these features, obtained from the trained caption bias model, are 0.4 and 0.7. Then, due to how the result was presented, it was $e^{1.1} \approx 3$ times as likely to be clicked than an equally relevant document presented at the same rank and average presentation (i.e., with medium-length URL and less highlighting). Then, the observed click is reweighted with the inverse of this change in click

[2]In contrast to previous chapters, we define the outcome in terms of the click difference, instead of the sign of the click difference. This formulation makes it easier to see how individual clicks are reweighted below. The resulting TD estimator (without bias compensation) is equivalent to the aggregation method called Δ_{click} in (Chapelle et al., 2012).

[3]While the change in click odds only approximates the change in click probability (relative risk) under caption bias, it has the advantage that it can be estimated from non-random samples using logistic regression. It is considered a good approximation of relative risk under the "rare disease assumption," in particular when the relative risk is small (Schmidt and Kohlmann, 2008). Because clicks are relatively rare and the odds ratios we observe are small, we consider these assumptions reasonable.

odds, to correct for the attractive presentation and decrease the click's contribution to the outcome of the interleaving experiment.

Applying the resulting click weights to the estimator for TD, and after substituting Eq. 5.2 in Eq. 4.1, we obtain the following alternative estimator, which reweights clicks to compensate for caption bias:

$$\hat{E}_{W-TD}[O] = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{r=1}^{len(\mathbf{l})}\frac{\mathbf{c}[r]}{e^{\beta_c\phi_c(\mathbf{l}[r],q)}}\mathbf{a}[r] - \sum_{r=1}^{len(\mathbf{l})}\frac{\mathbf{c}[r]}{e^{\beta_c\phi_c(\mathbf{l}[r],q)}}\bar{\mathbf{a}}[r]\right). \quad (5.3)$$

Thus, instead of weighting each observed click equally, this estimator for weighted TD weights each observed click by its change in click odds due to caption bias.

The corresponding estimator for PI-MA is obtained by substituting Eq. 5.2 in Eq. 4.5 and again applying click reweighting:

$$\hat{E}_{W-PI}[O] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{a}\in\mathbf{A}}\left(\sum_{r=1}^{len(\mathbf{l})}\frac{\mathbf{c}[r]}{e^{\beta_c\phi_c(\mathbf{l}[r],q)}}\mathbf{a}[r] - \sum_{r=1}^{len(\mathbf{l})}\frac{\mathbf{c}[r]}{e^{\beta_c\phi_c(\mathbf{l}[r],q)}}\bar{\mathbf{a}}[r]\right)P(\mathbf{a}|\mathbf{l}_i, q). \quad (5.4)$$

Our method for modeling caption bias is experimentally validated in the next section, and we apply the resulting model to interleaving experiments in §5.2.4.

5.2 Experiments and Results

In this section we detail our experimental setup and results in three parts. First, we focus on training caption bias models (5.2.1). Results of this step give insights into the features that were found to be useful for explaining click behavior and their relative importance. Second, we assess the quality of the trained models, by comparing their predictions to observed click behavior (5.2.2). Third, we show how our caption bias models can be applied to infer user preferences from clicks (5.2.3). Finally, we apply our best-performing caption-bias model to interleaving experiments and analyze its effect on experiment outcomes (5.2.4).

5.2.1 Modeling Caption Bias

In our first experiment, we train several instantiations of our caption bias model as defined in Equation 5.1. Analyzing which visual features contribute to explaining click behavior, and what their relative importance is in each of the models, allows us to better understand the observed caption bias. In this experiment, we consider the following instantiations of our model:

- **highlighting** - uses only pairwise highlighting features, similar to Yue et al. (2010b)
- **document-wise** - uses the per-document features in Table 5.1
- **pairwise** - uses only the pairwise features listed in Table 5.2
- **combined** - considers all visual features in Tables 5.1 and 5.2

All four models were trained using standard logistic regression (with the function glm) in R.[4] The training data was obtained from a large commercial web search engine. The data is a random sample of queries and result pages collected on February 16, 2012. To account for the effect of relevance, the log data was intersected with a large set of previously collected relevance judgments. This intersection resulted in approximately 420,000 (non-unique) query–URL pairs.

A limitation of our setup is that, because we use query–URL pairs for which relevance assessments were available, our training data set does not constitute a random sample. Rather, the use of previously collected judgments introduces bias, as more judgments were available for, e.g., frequent queries, and for documents that were previously ranked highly by the search engine. However, this bias only affects our training data, and not the data sets used for evaluation and analysis.

The weights for our trained models are shown in Table 5.3. Recall that when we apply our model of caption features to reweighting clicks, we only use the weights of the caption features to determine the relative change in click attractiveness, and ignore position and relevance features (cf., §5.2.1). Therefore, we only report and analyze the regression weights for these visual features.

For the *highlighting* model, we find a relatively weak effect for all included features. URL bolding has a positive effect (i.e., more bold increases the click likelihood of a result), but only when compared to the document ranked below. Increased highlighting in the title always increases click probabilities, and this effect is stronger than that of highlighting in the snippet (in agreement with (Yue et al., 2010b)). For increased highlighting in the snippet, a small negative effect is detected, which may be caused by easily identifiable caption spam.

For the *document-wise* model we identify several features that have a strong correlation with click behavior. The highest regression weight is obtained for our deep links feature. This result matches our observation that results with deep links tend to attract more clicks (even on the title link), perhaps because they take up more space on the result page. For highlighting in the result title, a much stronger effect is observed than for the pairwise version of this feature. Finally, click probability is found to decrease for URLs with many slashes, and for short snippets, as expected.

Results for the *pairwise* model are similar, although the trained weights are smaller in magnitude. For URL length, a negative impact is detected when the current result has a longer URL than the document above, however this effect is reversed when the URL is compared to the result below.

Finally, in our combined model we find that all per-document features found to be statistically significant previously again have a statistically significant impact, even when combined with pairwise features. However, significant effects are also found for additional pairwise features, suggesting that the click behavior observed in our training data can best be explained when document-wise and pairwise features are combined. The pairwise features that had a statistically significant effect when included in the combined model are Δ URL slashes below, Δ title length below, as well as all highlighting features for result title and snippet.

[4] Obtained from `http://www.r-project.org/`.

		highlighting	**document-wise**	**pairwise**	**combined**
Per-document features	Deep links		1.041 (±0.021)		1.048 (±0.021)
	Short URL		0.470 (±0.022)		0.403 (±0.022)
	URL slashes		-0.425 (±0.116)		-0.493 (±0.117)
	URL bold		0.314 (±0.028)		0.271 (±0.027)
	Short title		0.387 (±0.021)		0.264 (±0.023)
	Long title		0.353 (±0.021)		0.461 (±0.021)
	Title start		0.352 (±0.017)		0.292 (±0.017)
	Title bold		0.845 (±0.036)		0.672 (±0.037)
	Short snippet		-0.534 (±0.129)		-0.716 (±0.128)
	Long snippet		0.192 (±0.048)		0.324 (±0.049)
Pairwise features	Δ URL slashes above			-0.184 (±0.020)	
	Δ URL slashes below			0.160 (±0.015)	0.109 (±0.015)
	Δ URL bold above				
	Δ URL bold below	0.104 (±0.028)			
	Δ title length above (in characters)				
	Δ title length below (in characters)			-0.237 (±0.010)	-0.270 (±0.011)
	Δ title bold above (in words)	0.267 (±0.024)		0.305 (±0.024)	0.248 (±0.023)
	Δ title bold below (in words)	0.191 (±0.018)		0.260 (±0.018)	0.172 (±0.018)
	Δ snippet length above (in characters)			-0.159 (±0.010)	
	Δ snippet length below (in characters)				
	Δ snippet bold above (in sections)	-0.042 (±0.009)		-0.127 (±0.017)	-0.088 (±0.016)
	Δ snippet bold below (in sections)	-0.063 (±0.007)		-0.062 (±0.012)	-0.049 (±0.012)

Table 5.3: Results (modeling caption bias): Visual features included in each trained model, with estimated regression weights and standard error of the weight estimates. Of the candidate features for each model, only those with a statistically significant effect on click predictions are included (with $p < 0.001$).

To summarize, per-document features were found to be the most useful for predicting click behavior from visual characteristics. A weaker impact was identified for pairwise features, but a combined model best explains observed click behavior.

5.2.2 Evaluating Caption Bias Models

Above, we presented four models of caption bias given visual features. Here, we compare the performance of these models by applying the models to predict user clicks on a new data set.

The data for this experiment was again obtained from a commercial web search engine, but was collected several days after the training data. We obtained three non-overlapping random samples (by user), from February 23 to 26, 2012. Each data set consists of queries, the presented search results, and the observed clicks. Below, we refer to these evaluation sets as A, B, and C.

The task on which we evaluate the trained models of caption bias is to predict whether a given document will be clicked or not, based on its visual characteristics and its position in the result list. As ground truth, we use the actually observed clicks.

We measure performance in terms of perplexity, a measure that is typically used to compare the quality of the predictions of a probabilistic model to observed outcomes, e.g., to evaluate click prediction methods (Dupret and Piwowarski, 2008). It is formulated as $2^{-\sum_{i=1}^{n} \frac{1}{n} \log_2 P(c_i)}$, where c_i are observed events (here, whether a document was clicked or not), $P(c_i)$ is the probability of an observed event predicted by our model, and n is the number of observations. Intuitively, perplexity captures the degree of "surprise" that remains after a predictive model is applied. When applying an ideal model that can accurately predict all observed events, no surprise remains, and perplexity is 1. A uniformly random model would obtain a perplexity of 2, indicating that the model would not provide any information as to whether or not a result document is clicked.

In addition to the four models discussed in the previous section, we add a *baseline* model, that does not take any visual features into account (but is trained using relevance and position features, and predicts click behavior using document position alone). Our results are presented in Table 5.4.

data set	**baseline**	**highlighting**	**document-wise**	**pairwise**	**combined**
A	1.664▲	1.675▲	1.578▲	1.650▲	**1.552**
B	1.627▲	1.640▲	1.540▲	1.627▲	**1.521**
C	1.646▲	1.656▲	1.565	1.633▲	**1.540**

Table 5.4: Results (evaluating caption bias models): Perplexity of all caption bias models when predicting clicks. Statistical significance is indicated in comparison with the *combined* model.

Surprisingly, we find that the least predictive model is not the baseline (without any visual features), but the highlighting model. It performs worse than the baseline in all cases, even though it better explained click behavior on the training data. This suggests that the highlighting model may be overfitting the training data. The pairwise model is little better than the baseline, suggesting that pairwise features alone may not be able to accurately represent users' click decisions. Better performance is achieved by our document-wise model. The best performance (lowest perplexity) over all data sets is obtained by our combined model. Apart from data set C, where the document-wise model is not statistically different from the combined model, all other models on all other data sets perform significantly worse than our best-performing model. Our results suggest that a combined model of visual features, that takes both document-wise and pairwise features into account, is the most successful at modeling users' click decisions.

To better understand how well our combined model captures caption bias, we conduct a more detailed analysis of its performance on different segments of queries drawn from data set B. We analyze prediction performance by (1) query frequency, and (2) the type of the information need expressed by the query.

Table 5.5 shows the perplexity of the baseline and combined models, split by query frequency. We divide queries into three groups: *head*, which consists of the 20% most frequent queries, *body*, which consists of queries between the 20th and 80th frequency percentile, and *tail*, which consists of the 20% least frequent queries.

Segment	**baseline**	**combined**
Head	**1.105**▼	1.111
Body	1.400▲	**1.347**
Tail	2.057▲	**1.855**

Table 5.5: Perplexity of the baseline and combined models, split by query frequency segment. Statistical significance is indicated in comparison with the *combined* model.

We find that on *head* queries, perplexity is best for both models, and that the performance of the combined model is lower than that of the baseline. The reason is that these very frequent queries are typically "easy", because many users search for the same things, usually with a clearly identifiable goal. For this type of query, users are likely to recognize target result pages, e.g., by the URL. Thus, caption bias is low for this type of query, as reflected in our results. For the less frequent *body* queries, perplexity is higher for both models. Here, the performance of the combined model is significantly better than that of the baseline model, indicating that caption characteristics play a role here. The trend continues for the *tail* query segment. For this segment, the baseline performs worse than random, which indicates that result position is not a good predictor of clicks for these queries. The performance of the combined model is significantly better. The large improvement of 10% suggests that users' click decisions on this most difficult query segment are particularly strongly affected by caption characteristics.

Table 5.6 shows the results obtained when splitting queries by the type of information need. Here, we use two sources of information to categorize queries. First, we use information provided by the search engine used for data collection. For queries with one predominantly clicked result, this top result is given more space on the result page. Our log data provides information on which queries were treated with such "enhanced navigational" results, and we use queries marked as such as our first category. For the remaining queries, a simple click-entropy based classifier divides queries into navigational and non-navigational.

Segment	baseline	combined
Enhanced Navigational	**1.143** ▼	1.149
Navigational	1.357	**1.339**
Non-navigational	1.924 ▲	**1.744**

Table 5.6: Perplexity of the baseline and combined models, split by the type of information need. Statistical significance is indicated in comparison with the *combined* model.

We observe a similar pattern to that obtained using query frequency. For *enhanced navigational* queries, perplexity is lowest, and the baseline model performs better than the combined model. As for the *head* queries above, results for these queries are the least likely to suffer from caption bias. These queries have one main target result, and this result is easy to identify on the result page, so caption bias is expected to play a small role here. For both navigational and non-navigational queries perplexity is higher for both models but the combined model performs significantly better in both cases. Again, the performance improvement of the combined over the baseline model is largest for non-navigational queries, where caption bias is expected to be strongest.

To summarize, we validated our models of caption bias on the task of predicting clicks on interleaved result lists. Our combined model, which includes both document-wise and pairwise features to model caption bias, performed best overall. We also found that our model worked best on queries where caption bias is expected to affect clicks the most, namely infrequent, non-navigational queries.

5.2.3 Predicting Preferences

In the previous subsections, we presented our results for training and evaluating caption bias models, and found that our combined model predicted observed click behavior best. In this section, we show how applying this model to reweight clicks affects interleaving scores on individual results, and show how such a reweighting can be used to predict user preferences.

To show how compensating for caption bias affects interleaved comparisons in detail, consider Table 5.7. For the query “today in two minutes”, four search results are shown. The first has a visual representation that results in a weight of 1.003, which is close to the average (i.e., the result is about as likely to be clicked as a result for which all visual features are false / zero). Click probability increases, e.g., due to the short title, and highlighting in the URL, but decreases due to the missing snippet and the lack of highlighting in the title. Overall, the result is relatively unlikely to be clicked based on attractiveness alone. In contrast, the lower-ranked results look more attractive, and consequently receive lower click weights. In this result list, the two results with the lowest weights (i.e., the most attractive visually) were clicked by the user. This suggests that click behavior may have been affected by caption bias.

We can now compare the outcomes that would be obtained in the above example when inferring a TD outcome with and without applying our model of caption bias.

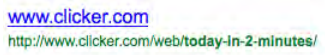
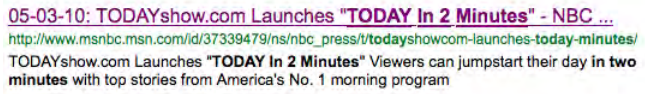
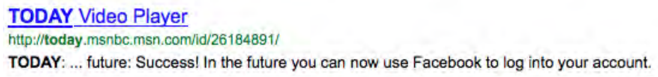
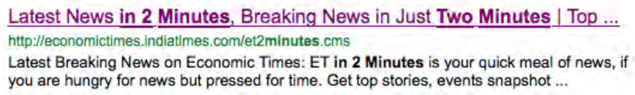
Rank	Example results for the query "today in two minutes"	Visual features	Ranker	Weight
1	 www.clicker.com http://www.clicker.com/web/**today-in-2-minutes**/	URL bold, short title, short snippet, Δ URL slashes below $= -1$, Δ title length below $= -1$, Δ title bold below $= -1$, Δ snippet length below $= -1$, Δ snippet bold below $= -1$	B	1.003
2*	 05-03-10: TODAYshow.com Launches "**TODAY In 2 Minutes**" - NBC ... http://www.msnbc.msn.com/id/37339479/ns/nbc_press/t/**todayshow**com-launches-**today-minutes**/ TODAYshow.com Launches "**TODAY In 2 Minutes**" Viewers can jumpstart their day **in two minutes** with top stories from America's No. 1 morning program	URL slashes, URL bold, long title, title bold, Δ title bold above $= 1$, Δ snippet bold above $= 1$	A	0.343
3	 **TODAY** Video Player http://**today**.msnbc.msn.com/id/26184891/ **TODAY**: ... future: Success! In the future you can now use Facebook to log into your account.	short title, title bold, Δ URL slashes below $= 1$, Δ title length below (in characters) $= -1$, Δ title bold below $= -1$, Δ snippet length below $= -1$, Δ snippet bold below $= -1$	B	0.517
4*	 Latest News **in 2 Minutes**, Breaking News in Just **Two Minutes** | Top ... http://economictimes.indiatimes.com/et2**minutes**.cms Latest Breaking News on Economic Times: ET **in 2 Minutes** is your quick meal of news, if you are hungry for news but pressed for time. Get top stories, events snapshot ...	long title, title bold, Δ title bold above $= 1$, Δ snippet bold above $= 1$	A	0.274

Table 5.7: Example interleaving impression with visual features, contributing ranker, and click weight. The ranks of clicked results are marked with '*' and corresponding result titles are highlighted in purple.

We observe that both clicked results were contributed by ranker A, leading to a win of 2 clicks over B. When applying the caption bias model, the low click weights of the clicked results are taken into account, leading to a much smaller win of 0.617. This example shows how the caption bias model decreases the impact of clicks that may have been biased towards more attractive results.

As an additional proof-of-concept, we applied our caption-bias weighting scheme to a small sample of search result impressions for which two different URLs had been clicked by different users. When a model predicted a lower weight for one of the clicked URLs, this URL would be inferred to be more likely to be clicked due to its presentation, and the URL with the higher click weight would be inferred to be preferred due to its content. For this small data set, we asked human annotators to judge which of two landing pages (for the two competing URLs) they would prefer for a given query.

Table 5.8 shows how often our model predictions agree with the human preference judgments. For the baseline model, no preferences can be inferred, as both URLs were clicked for the given query. Prediction quality of the pairwise model is the same as a random model would achieve, while accuracy for the document-wise and highlighting models are slightly higher. The best preference predictions are obtained by our combined model.

baseline	**highlighting**	**document-wise**	**pairwise**	**combined**
0 / 86 / 0	44 / 1 / 41	43 / 11 / 32	43 / 0 / 43	**47 / 0 / 39**

Table 5.8: Preference predictions by caption bias models. For each model we include the number of correct / no preference / incorrect predictions.

5.2.4 Interleaving Outcomes

In this section, we investigate whether and how interleaving outcomes can be affected by differences in result presentation in practice. To this end, we apply our best caption bias model (the combined model, as shown in Table 5.3) to interleaved comparisons conducted on live web search traffic using the reweighting schemes introduced in §5.1.3, and analyze how the inferred interleaving outcomes are affected by bias compensation.

We conducted six interleaving experiments (referred to as **E1**-**E6** below), selecting experiments that represented small changes in ranking quality that are typical of incremental ranker improvements at major web search engines. We also selected pairs such that the competing rankers used methods for applying previously collected clickthrough data, and different weights to make the influence of clicks weaker or stronger. Selecting ranker pairs in this way increased our chances of detecting changes in interleaving outcomes due to caption bias. If caption bias affects interleaving outcomes beyond increasing noise, then we expect that compensating for caption bias results in outcome changes of different magnitudes for these experiments. On the other hand, if caption bias affects all rankers equally, then all experiment outcomes should be affected by bias compensation equally. The direction of all experiments is chosen such that a *baseline* ranker ($\mathbf{l}_1$) is compared to a *treatment* ranker ($\mathbf{l}_2$). The hypothesis for all six interleaving experiments is that the treatment ranker improves over the baseline ranker.

The interleaving experiments were run on a sample of live traffic from the same web search engine as used in the previous two sections. These experiments were run within three weeks of collecting the data sets for model training and evaluation, so that no major changes in click behavior are expected. Interleaved result lists were generated using TD.[5] After observing user clicks, four different scoring methods were applied to compute interleaved comparison outcomes using the original TD (Eq. 4.1) and PI-MA (Eq. 4.5) scoring schemes (without compensating for caption bias) and our caption-bias reweighting schemes (Eqs. 5.3–5.4) (with the combined caption bias model). Note that in earlier work, click preferences were recentered to let a value of 0.5 denote "no preference" (i.e., the rankers are inferred to perform equally well), as shown in Eq., 4.2 (defined on page 42). Then, the treatment ranker is detected to win the comparison if the estimated outcome is statistically significantly higher than 0.5. Here we follow the same convention, and report all scores centered around 0.5.

Table 5.9 gives an overview of the interleaved comparison outcomes obtained with TD and PI-MA, before and after caption bias reweighting. We can see that scores are generally close to 0.5. These scores reflect the incremental ranker changes typically tested at major web search engines (e.g., changes that affect a small percentage of queries). Despite the relatively small changes in ranking, most experiments detect a statistically significant difference between the rankers.

Experiment	**TD**		**PI-MA**	
	Unweighted	**Weighted**	**Unweighted**	**Weighted**
E1	$0.5033^{\blacktriangle}$	$0.5017^{\blacktriangle}$	$0.5018^{\blacktriangle}$	$0.5011^{\blacktriangle}$
E2	$0.5040^{\blacktriangle}$	$0.5020^{\blacktriangle}$	$0.5024^{\blacktriangle}$	$0.5016^{\blacktriangle}$
E3	$0.5017^{\blacktriangle}$	$0.5010^{\blacktriangle}$	$0.5008^{\blacktriangle}$	$0.5006^{\blacktriangle}$
E4	$0.5031^{\blacktriangle}$	$0.5005^{\blacktriangle}$	$0.5010^{\blacktriangle}$	0.5000
E5	$0.5018^{\blacktriangle}$	$0.5014^{\blacktriangle}$	$0.5008^{\blacktriangle}$	$0.5008^{\blacktriangle}$
E6	0.4999	$0.4998^{\blacktriangledown}$	0.5000	$0.4999^{\blacktriangledown}$

Table 5.9: Results (interleaving): Interleaving scores using TD (PI-MA) before (*unweighted*, Eqs. 4.1 and 4.5) and after (*weighted*, Eqs. 5.3 and 5.4) applying caption-bias models to six interleaved comparison experiments. Outcomes marked with $^{\blacktriangle}$ or $^{\blacktriangledown}$ detected significant gains or losses of the treatment ranker in comparison with the original ranker.

We first compare the outcomes obtained under TD and PI-MA (cf., the "unweighted" columns in Table 5.9). Outcomes for all experiments agree in direction and in whether the

[5]As a result, the implementation of PI-MA used here is a variant of the method described in §4.2.2. In particular, we use the same interleaving process as TD (that is, we do not interleave probabilistically) to minimize effects on user experience. However, we compute comparison outcomes as if PI had been used for interleaving. This results in comparisons that weight clicks by the magnitude of the difference in position between rankers. For example, a ranker would gain a small win (in terms of weighted click) for moving a document up by one rank, and a large win when the clicked document was moved up from a much lower rank.

difference is statistically significant. However, the magnitude of outcomes differs. E.g., the change in comparison outcomes between TD and PI-MA smaller for E1 than for E4. The reason is that, under TD, individual comparison outcomes are binary, independent of the magnitude of the difference between rankers. Under PI-MA, outcomes are weighted by the rank distance of clicked documents. E.g., if a click was observed on a document placed at ranks 3 and 4 by the competing rankers, the magnitude of the interleaving outcome would be much smaller than if one ranker had placed the document at rank 1 and the other at rank 10. Thus, for experiments where there is a large absolute difference between TD and PI-MA outcomes, a large portion of the differences detected under TD is expected to be caused by relatively small differences between rankings (e.g., E4).

After applying reweighting (cf., the "weighted" columns in Table 5.9), we find that the detected interleaving gains are generally smaller than under the original scoring methods. This suggests that a portion of the observed clicks was on results with low weights (i.e., with high click probability). In most cases, experiments for which significant differences between rankers were detected before reweighting, are still significantly different after reweighting (E1, E2, E3, E5). However, most importantly, we see differences in the strength of the effects of reweighting.

One example where comparison outcomes appear to be strongly affected by caption bias is ranker pair E4. For this pair, the original comparison using TD results in a statistically significant gain for the treatment ranker (0.5031, a relatively big difference in typical ranker evaluations). After reweighting, a much smaller (but still significant) improvement of 0.5005 is observed, indicating that most of the gain observed under TD may be due to caption bias. Comparing to the outcome obtained under the probabilistic method, we find that another portion of observed improvements is due to only small ranking changes. After our model of caption bias is applied to PI-MA, no difference between the rankers can be detected. This suggests that the initially detected improvement was due to small ranking changes and caption bias, and that there is no true improvement in ranker quality. We further analyzed the ranker pairs used in our interleaving experiments, and found that the treatment ranker in E4 relied on click data the most. This suggests that this experiment is particularly likely to be affected by caption bias.

For experiment E6, the comparison outcome changes from non-significant to significantly worse when caption bias reweighting is applied. Here, the original comparison would support the conclusion that the compared rankers are equivalent. However, the reweighted outcome indicates that the treatment ranker was really significantly worse than the baseline ranker, when rank distances and caption bias are taken into account.

Our results support the hypothesis that caption bias can affect the outcomes of interleaving experiments. The assumption that caption bias may affect both rankers equally, leading to a mere increase in noise, is not supported, because our experiments showed different behaviors when caption bias was compensated for. In experiments E1, E2, E3, and E5 the direction of the interleaving outcomes and their statistical significance were maintained after applying caption bias reweighting. In E4 and E6 the inferred outcomes changed, in E4 (where the treatment ranker strongly relied on click signals) from a significant improvement to a tie, in E6 from a tie to a significant loss for the treatment ranker. We conclude that caption bias can affect the outcomes of interleaved comparison experiments, and that, if caption bias is not compensated for, it can lead to drawing the wrong conclusions about the relative quality of result rankings.

5.3 Conclusion

In this chapter, we addressed the problem of caption bias in interleaving experiments. Interleaved comparison methods promise to capture user preferences, because they rely on interactions of actual users. However, when click behavior is systematically biased by, e.g., the visual appearance of search results, interleaved comparison methods may detect differences between rankers that are not related to true ranker quality in terms of document relevance, or they may fail to detect true differences between rankers. Here, we presented models of caption bias, and investigated how caption bias can affect click behavior and interleaving outcomes.

We started our investigation of caption bias by introducing a set of models designed to model bias using per-document features that capture visual characteristics of individual result documents, and pairwise features that capture relationships with neighboring documents. We evaluated these models by using them to predict clicks. We found that overall, per-document features were more successful in capturing click behavior than pairwise features. However, best results were achieved using a combined model that uses both feature sets. We also found that the combined caption bias model was the most successful at predicting click behavior in cases where caption bias is expected to be strongest (such as non-navigational queries). Finally, the combined model was the most accurate in predicting judged preferences between pairs of documents. We conclude that the appearance of document captions and neighboring captions significantly affects users' click behavior.

We derived two extensions of the interleaved comparison methods TD and PI-MA to integrate probabilistic caption bias models such as the one devised in this chapter to compensate for caption bias. We showed that, when caption bias can be modeled accurately, integrating the resulting model with interleaved comparison methods is possible and leads to unbiased estimates of comparison outcomes.

Finally, we applied our best (combined) caption bias model to six interleaving experiments conducted on live search traffic of a major commercial web search engine. We found that compensating for caption bias led to small changes in all experiment outcomes. Most importantly, there were differences in the magnitude of the effects. In one experiment, an originally large and statistically significant difference between rankers was nullified after rank differences and caption bias were taken into account. The outcome of a second interleaving experiment changed from "not significant" to detecting a significant loss for the treatment ranker. Our results show that the outcomes of interleaving experiments can be affected by caption bias, and that without compensating for caption bias wrong conclusions can be drawn.

The results of this chapter impact the research in online evaluation and online learning to rank as follows. While work on interleaved comparison methods was based on the assumption that caption bias would affect rankers in an interleaving experiment equally, thus leading to noise but not bias, we found that this is not always the case. With only a small number of interleaving experiments we were able to identify cases where interleaved comparison outcomes changed when caption bias was compensated for. However, we also showed that probabilistic models of caption bias can be integrated with interleaved comparison methods to compensate for caption bias. This means that in practice, unbiased interleaved comparisons are possible and can be used for online evaluation and learning to rank for IR.

The models of caption bias developed in this chapter were shown to predict click behavior more accurately than models without caption features. However, they constitute only a first step towards capturing the complex effects that visual aspects of search engine result pages may have on click behavior. An important direction for future work is to develop more accurate models of caption bias, possibly taking into account recent work on click modeling (Dupret and Piwowarski, 2008). Particularly, models that can generalize across queries (Zhu et al., 2010), and that separate perceived relevance from judged relevance (Zhong et al., 2010) are promising in this context.

In this and the previous chapters we focused on interleaved comparison methods as a way of inferring feedback from natural user interactions in an online setting. These methods are important by themselves, for online evaluation experiments, e.g., to evaluate new retrieval technologies. However, in this thesis we are mainly interested in these methods as a component of online learning to rank for IR systems. In Chapter 6, we will focus on the principles that allow learning from interleaved comparisons. We also compare them to approaches that learn from document-pairwise feedback.

6 Balancing Exploration and Exploitation

In the previous two chapters we developed and investigated interleaved comparison methods as a promising solution for inferring information about rankers from implicit user feedback. In this and the next chapter, we focus on the question of how to learn reliably and efficiently from the inferred feedback.[1]

Methods for online learning to rank for IR need to address a number of challenges. First, the most robust methods for inferring feedback provide only relative information, e.g., about the relative quality of documents (§2.3) or rankers (see the interleaved comparison approaches discussed in previous chapters). Algorithms for learning from such relative feedback have been proposed (§2.5), and these form our baseline algorithms. Second, even relative feedback can be noisy and biased. Our empirical results in this chapter provide first insights into how algorithms for learning to rank from pairwise and listwise relative feedback perform under noise.

A challenge in online learning to rank for IR that has not been addressed previously is that algorithms for this setting need to take into account the effect of learning on users. In contrast to offline approaches, where the goal is to learn as effectively as possible from the available training data, online learning affects, and is affected by, how user feedback is collected. Ideally, the learning algorithm should not interfere with the user experience, observing user behavior and learning in the background, so as to present search results that meet the user's information needs as well as possible at all times. This would imply passively observing, e.g., clicks on result documents. However, passively observed feedback can be biased towards the top results displayed to the user (Silverstein et al., 1999). Learning from this biased feedback may be suboptimal, thereby reducing the system's performance later on. Consequently, an online learning to rank approach should take into account both the quality of current search results, and the potential to improve that quality in the future, if feedback suitable for learning can be observed.

In this chapter, we frame this fundamental trade-off as an *exploration–exploitation*

[1]This chapter is based on work presented in Hofmann et al. (2011a, 2013b). However, the empirical results presented here differ from our previous work as follows. First, the *navigational* click model is instantiated differently (as shown in §3.3) to match the model used in Chapter 4. Second, we previously applied our algorithms to the documents provided by the LETOR data sets in their original order. Because this order is non-randomized, learners started from high-quality lists, which could result in higher absolute performance. In the experiments presented here, we address this problem by breaking ties randomly (so that a result list generated from a zero weight vector is completely randomized). Despite these changes, the results are qualitatively identical to those presented earlier and support the same conclusions.

dilemma. If the system presents only document lists that it expects will satisfy the user, it cannot obtain feedback on other, potentially better, solutions. However, if it focuses too much on document lists from which it can gain a lot of new information, it risks presenting bad results to the user during learning. Therefore, to perform optimally, the system must *explore* new solutions, while also maintaining satisfactory performance by *exploiting* existing solutions. Making online learning to rank for IR work in realistic settings requires effective ways to balance exploration and exploitation.

We investigate mechanisms for achieving a balance between exploration and exploitation when using pairwise and listwise methods, the two most successful approaches for learning to rank in IR (§2.2). The pairwise approach takes as input pairs of documents with labels identifying which is preferred and learns a classifier that predicts these labels. In principle, pairwise approaches can be directly applied online, as preference relations can be inferred from clicks. However, as we demonstrate in this chapter, balancing exploration and exploitation is crucial to achieving good performance.

Listwise approaches aim to directly optimize an evaluation measure, such as NDCG, that concerns the entire document list. Since such evaluation measures cannot be computed online, new approaches that work with implicit feedback have been developed (Yue and Joachims, 2009). The existing algorithm learns directly from the relative feedback that can be obtained from interleaved comparison methods, but we show that it over-explores without a suitable balance of exploration and exploitation.

We present the first two algorithms that can balance exploration and exploitation in settings where only relative feedback is available. First, we start from a pairwise approach that is initially purely exploitative (§2.5.1). Second, we start from a recently developed listwise algorithm that is initially purely exploratory (Yue and Joachims, 2009) (§2.5.2). We assess the resulting algorithms using the evaluation framework described in Chapter 3 to answer the following questions:

RQ 11 Can balancing exploration and exploitation improve online performance in online learning to rank for IR?

RQ 12 How are exploration and exploitation affected by noise in user feedback?

RQ 13 How does the online performance of different types (pairwise and listwise) of online learning to rank for IR approaches relate to balancing exploration and exploitation?

Our main result is that finding a proper balance between exploration and exploitation can substantially and significantly improve the online retrieval performance in pairwise and listwise online learning to rank for IR. In addition, our results are the first to shed light on the strengths and weaknesses of pairwise and listwise learning in an online setting, as these types of approaches have previously only been compared offline. We find that learning from document-pairwise feedback can be effective when this feedback is reliable. However, when feedback is noisy, a high amount of exploration is required to obtain reasonable performance. When clicks are interpreted as listwise feedback, learning is similarly effective as under the pairwise interpretation, but it is much more robust to noise. However, online performance under the original listwise learning approach is suboptimal, as it over-explores. Dramatically reducing exploration allows learning rankers equally well, but at much lower cost. Consequently, balancing exploration and exploitation in the listwise setting results in significantly improved online performance under all levels of noise. We discuss in detail the effects on each approach of balancing exploration and exploitation, the amount of noise in user feedback, and characteristics of the data sets. Finally, we describe the implications of our results for making these approaches work effectively in practice.

The remainder of this chapter is organized as follows. We present our methods for balancing exploration and exploitation in the pairwise and listwise setting in §6.1. Our experiments are described in §6.2, followed by results and analysis in §6.3. We conclude in §6.4.

6.1 Approaches

In this section, we describe our approaches for balancing exploration and exploitation for learning to rank in IR. These build on the pairwise and listwise baseline learning algorithms shown in §2.5. Our approaches are based on the problem formulation of online learning to rank for IR as a contextual bandit problem, as shown in §3.1.

6.1.1 Balancing Exploration and Exploitation in Pairwise Learning to Rank

Our first approach builds off a pairwise formulation of learning to rank (Herbrich et al., 1999; Joachims, 2002), in particular the stochastic gradient descent algorithm presented in Sculley (2009). As detailed in §2.5.1, this algorithm optimizes a weight vector $\mathbf{w}$ for linear combinations of ranking features $\mathbf{x} = \phi(d, q)$ to minimize a loss function formulated in terms of the pairwise ranking loss (see Algorithm 4 on page 29). As shown by Joachims (2002), the required pairwise feedback can be inferred from implicit feedback, such as click data.

In previous applications of pairwise learning to implicit feedback scenarios, learning was performed in a batch setting. First, implicit feedback was collected given an initial ranking function. Then, the algorithm was trained on all collected implicit feedback. Finally, this trained system was deployed and evaluated (Joachims, 2002). In this setting, data collection is naturally exploitative. Users are shown results that are most likely to be relevant according to a current best ranking function. In the online setting, such an exploitative strategy is expected to result in the highest possible short-term performance. However, it is also expected to introduce bias, as some documents may never be shown to the user, which may result in sub-optimal learning and lower long-term performance. This is confirmed in our experiments, as we will see below.

In supervised applications of pairwise learning to rank methods, the learning algorithm is typically trained on the complete data set. Sculley (2009) developed a sampling scheme that allows the training of a stochastic gradient descent learner on a random subset of the data without a noticeable loss in performance. Document pairs are sampled randomly such that at each learning step one relevant and one non-relevant document were selected to form a training pair. In the online setting, we expect such a fully exploratory strategy to result in minimal training bias and best long-term learning.

Algorithm 10 Balancing exploration and exploitation in the pairwise setting.

```
1: Input: $\mathcal{D}$, $\eta$, $\lambda$, $\mathbf{w}_0$, $\epsilon$
2: for query $q_t$ ($t = 1..\infty$) do
3:     $\mathbf{X} = \phi(\mathcal{D}|q_t)$                                  // extract features
       // generate exploitative result list
4:     $S = \mathbf{w}_{t-1}^T \mathbf{X}$
5:     $\mathbf{l}_1 = sort\_descending\_by\_score(\mathcal{D}, \mathbf{s})[1:10]$
6:     $\mathbf{l}[r] \leftarrow$ first element of $\mathbf{l}_1 \notin \mathbf{l}$ with probability $\epsilon$; element randomly sampled with-
       out replacement from $\mathcal{D} \setminus \mathbf{l}$ with probability $1 - \epsilon$
7:     Display $\mathbf{l}$ and observe clicked elements $\mathbf{c}$.
8:     Construct all labeled pairs $\mathcal{P} = (\mathbf{x}_1, \mathbf{x}_2, y)$ for $q_t$ from $\mathbf{l}$ and $\mathbf{c}$.
9:     for $(\mathbf{x}_1, \mathbf{x}_2, y)$ in $\mathcal{P}$ do
10:        if $y\mathbf{w}_{t-1}^T(\mathbf{x}_1 - \mathbf{x}_2) < 1.0$ and $y \neq 0.0$ then
11:            $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y(\mathbf{x}_1 - \mathbf{x}_2) - \eta\lambda\mathbf{w}_{t-1}$          // update $\mathbf{w}_t$
```

In the online setting where we learn from implicit feedback, we cannot directly determine for which document pairs we obtain feedback from the user. Any document list that is presented in response to a query may result in zero or more clicks on documents, such that zero or more pairwise constraints can be extracted. Due to position bias (Silverstein et al., 1999), the higher a document is ranked in the result list presented to the user, the more likely it is to be inspected and clicked.

Here, we ignore explicit dependencies between displayed documents, and define two document lists, one exploratory and one exploitative, that are then combined to balance exploration and exploitation. The exploitative list is generated by applying the learned weights to compute document scores and then sorting by score, as in the baseline algorithm. The exploratory list is generated by uniform random sampling of the documents associated with a query.[2]

We employ a method for balancing exploration and exploitation that is inspired by *ϵ-greedy*, a commonly used exploration strategy in RL (§2.4.2).[3] The difference between our approach and ϵ-greedy is that we do not pick a single action at each timestep, but rather select a number of actions that are presented simultaneously. This results in Algorithm 10, which differs from our baseline algorithm in how the result list is constructed (line 6).

We vary the relative number of documents from the exploratory and exploitative lists as determined by $\epsilon \in [0, 1]$. For each rank, an exploitative action (a document from the exploitative list) is selected with probability $1 - \epsilon$. A document from the exploratory list

[2]In practice, candidate documents are typically collected based on some feature-based criteria, such as a minimum score. Here, we use the candidate documents provided with the learning to rank data sets used in our experiment, where candidate selection may have been biased (Minka and Robertson, 2008). However, bias in terms of feature values can be neglected here, as the specifics of the learned ranker are not the subject of this study, and all learning methods are affected equally.

[3]More complex schemes of balancing exploration and exploitation are of course possible, but our focus here is on demonstrating the benefit of such a balance over purely exploratory and purely exploitative forms of soliciting feedback. A simple scheme is sufficient for this goal. We also experimented with a more complex softmax-like algorithm and obtained qualitatively similar results. However, such an algorithm is more difficult to tune than the ϵ-greedy-like algorithm used here (Sutton and Barto, 1998; Whiteson and Stone, 2006a).

is selected with probability ϵ. Thus, values of ϵ close to 0 mean that little exploration is taking place, making the algorithm collect feedback in an exploitative way ($\epsilon = 0$ corresponds to the purely exploitative baseline setting). Values close to 1 mean more exploration.

6.1.2 Balancing Exploration and Exploitation in Listwise Learning to Rank

Our second online learning to rank approach builds off DBGD (Yue and Joachims, 2009). This algorithm has been specifically developed for learning to rank in an online setting, and it requires only relative evaluations of the quality of two document lists and infers such comparisons from implicit feedback (Radlinski et al., 2008b). An overview of DBGD is given in §2.5.2.

Given an appropriate function for comparing document lists, DBGD learns effectively from implicit feedback. However, the algorithm always explores, i.e., it constructs the result list in a way that minimizes bias between the exploratory and exploitative document lists, which is assumed to produce the best feedback for learning. We now present a comparison function $f(\mathbf{l}_1, \mathbf{l}_2)$ that does allow balancing exploration and exploitation.

We base our comparison method $f(\mathbf{l}_1, \mathbf{l}_2)$ on BI (Joachims, 2003; Radlinski et al., 2008b), as detailed in §2.3.1 (in particular, Algorithm 1 on page 20). Extending this algorithm to balance exploration and exploitation is easiest compared to other interleaved comparison methods, and this is sufficient to test our hypothesis that balancing exploration and exploitation in online learning to rank for IR can improve online performance. Algorithms for balancing exploration and exploitation based on other interleaved comparison methods are possible and will be investigated in the future. A related approach, which is based on PI-MA and PI-MA-IS (Chapter 4) and improves online performance by reusing previously collected interaction data for more effective exploration, is presented in Chapter 7.

In contrast to previous work, we alter BI to randomize not only the starting list and then interleaving documents deterministically, but instead we randomly select the list to contribute the document at each rank of the result list. In expectation, each list contributes documents to each rank equally often. We call this altered version of BI *stochastic BI*.

Constructing result lists using stochastic BI allows us to apply a method similar to ϵ-greedy. The resulting algorithm, which supplies the comparison method that is required by DBGD, is shown in Algorithm 11. The algorithm takes as input two document lists $\mathbf{l}_1$ and $\mathbf{l}_2$, and an exploration rate k. For each rank of the result list to be filled, the algorithm randomly picks one of the two result lists (biased by the exploration rate k). From the selected list, the highest-ranked document that is not yet in the combined result list is added at this rank. The result list is displayed to the user and clicks $\mathbf{c}$ are observed. Then, for each clicked document, a click is attributed to list $\mathbf{l}_i$ ($i \in \{1, 2\}$) if the document is in the top v of $\mathbf{l}_i$, where v is the lowest-ranked click (as in Algorithm 1).

The exploration rate $k \in [0.0, 0.5]$ controls the relative amount of exploration and exploitation, similar to ϵ. It determines the probability with which a list is selected to contribute a document to the interleaved result list at each rank. When $k = 0.5$, an

Algorithm 11 $f(\mathbf{l}_1, \mathbf{l}_2)$ – k-greedy comparison of document lists using stochastic BI.

```
1: Input: $\mathbf{l}_1, \mathbf{l}_2, k$
2: $\mathbf{l} = [], n_1 = 0; n_2 = 0$
3: while $(len(\mathbf{l}) < len(\mathbf{l}_1)) \wedge (len(\mathbf{l}) < len(\mathbf{l}_2))$ do
4:     $a \leftarrow 1$ with probability $k$ else $2$
5:     $j = min\{i : \mathbf{l}_a[i] \notin \mathbf{l}\}$
6:     $append(\mathbf{l}, \mathbf{l}_a[j])$
7:     $n_a = n_a + 1$
   // present $\mathbf{l}$ to user and observe clicks $\mathbf{c}$, then infer outcome (if at least one click was observed)
8: $d_{max}$ = lowest-ranked clicked document in $\mathbf{l}$
9: $v = min\{j : (d_{max} = \mathbf{l}_1[j]) \vee (d_{max} = \mathbf{l}_2[j])\}$
10: $c_1 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_1[1..v]\}$
11: $c_2 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_2[1..v]\}$
   // compensate for bias (Eq. 6.1)
12: $c_2 = \frac{n_1}{n_2} * c_2$
13: return $-1$ if $c_1 > c_2$ else $1$ if $c_1 < c_2$ else $0$
```

equal number of documents are presented to the user in expectation.[4] As k decreases, more documents are contributed by the exploitative list, which is expected to improve the quality of the result list but produce noisier feedback.

As k decreases, more documents from the exploitative list are presented, which introduces bias for inferring feedback. The bias linearly increases the expected number of clicks on the exploitative list and reduces the expected number of clicks on the exploratory list. We can partially compensate for this bias since

$$E[c_2] = \frac{n_1}{n_2} * E[c_1], \tag{6.1}$$

where $E[c_i]$ is the expected number of clicks within the top v of list $\mathbf{l}_i$, and n_i is the number of documents that $\mathbf{l}_i$ contributed to the interleaved result list. This compensates for the expected number of clicks, but some bias remains, because the observed clicks are converted to binary preference decisions before they are aggregated over queries. While perfectly compensating for bias is possible, it would require making probabilistic updates based on the observed result. This would introduce additional noise, creating a bias-variance trade-off. Preliminary experiments show that the learning algorithm is less susceptible to increased bias than to increased noise. Therefore we use this relatively simple, robust bias correction. More complex, unbiased sampling schemes can be developed using PI-MA and PI-MA-IS (Chapter 4), but this is beyond the scope of this thesis.

[4]Note that the setting $k = 0.5$ corresponds to the fully exploratory baseline algorithm. Setting $k > 0.5$ would typically not increase the amount of information that can be gained from a comparison, but would hurt the expected reward, because fewer exploitative documents would be shown.

6.2 Experiments

In this section, we describe the experiments that evaluate the algorithms presented in §6.1. All experiments use the experimental setup detailed in Chapter 3. Here, we provide further details and give an overview of the experimental runs we evaluate in the pairwise and listwise setting.

For the experiments in this chapter we use the 9 LETOR 3.0 and 4.0 data sets (§3.4). Click data is generated using the *perfect*, *navigational*, *informational*, and *almost random* click models as shown in Table 3.1.[5] As detailed in §3.5, we use the training folds of each data set for training during the learning cycle and for calculating online performance (in terms of discounted cumulative NDCG, with $\gamma = 0.995$). We use the test sets for measuring final performance (in terms of NDCG).

For each data set we repeat all runs 25 times and report results averaged over folds and repetitions. We test for significant differences with the baseline runs (purely exploitative for the pairwise approach ($\epsilon = 0.0$), purely exploratory for the listwise approach ($k = 0.5$)) using a two-sided student's t-test (§3.5).

6.2.1 Pairwise Approach

In all pairwise experiments, we initialize the starting weight vector $\mathbf{w}_0$ to zero. In preliminary experiments we evaluated offline performance for $\eta \in \{0.0001, 0.001, 0.01, 0.1\}$, and selected the setting that performed best over all data sets ($\eta = 0.001$). Our *baseline* is the pairwise formulation of learning to rank with stochastic gradient descent as described in §6.1.1, in the fully exploitative setting ($\epsilon = 0$; equivalent to Algorithm 4). Against this baseline we compare increasingly exploratory versions of the algorithm ($\epsilon \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$). All experiments are run for 1,000 iterations.

6.2.2 Listwise Approach

In all listwise experiments, we initialize the starting weight vector $\mathbf{w}_0$ to zero. We use the best performing parameter settings from (Yue and Joachims, 2009): $\delta = 1$ and $\alpha = 0.01$ (these settings resulted in good performance over all data sets in our preliminary experiments). Our *baseline* is Algorithm 5, based on (Yue and Joachims, 2009), which corresponds to a purely exploratory setting of $k = 0.5$ in our extended method.[6] Against this baseline we compare *exploit* runs that balance exploration and exploitation by varying the exploration rate k between 0.4 and 0.1 as shown in Algorithm 11. Again, we run all experiments for 1,000 iterations.

6.3 Results and Discussion

In this section, we present the results of our experiments, designed to test our main hypothesis — that balancing exploration and exploitation can improve the online perfor-

[5]Results for the *almost random* click model are omitted, as they did not result in any new insights beyond those obtained from the other three click models.

[6]In the listwise approach, the highest level of exploration is reached when the two candidate lists are interleaved in equal parts, i.e., $k = 0.5$.

mance of online learning to rank for IR systems. We first address this hypothesis for the pairwise learning algorithm (§6.3.1), and then for the listwise learning algorithm (§6.3.2). For both approaches we further analyze online and offline performance to identify factors that affect online performance. Finally, we compare the two approaches under the novel perspective of balancing exploration and exploitation (§6.3.3).

6.3.1 Pairwise Learning

We present our results for the experiments on the pairwise approach, described in §6.2.1, in Table 6.1. It shows the online performance of the baseline approach (*exploitative*, $\epsilon = 0$) and increasingly more exploratory runs ($\epsilon > 0.0$) for the 9 LETOR 3.0 and 4.0 data sets and the *perfect*, *navigational*, and *informational* click models.

We expect good online performance for the exploitative baseline if the algorithm can learn well despite any bias introduced due to the high level of exploitation. Generally, an online learning to rank approach should exploit as much as possible, as it ensures that users see the best possible result lists given what has been learned. However, if increased exploration results in sufficiently high gains in offline performance, its short-term cost may be outweighed by its long-term benefits, as it increases the quality of result lists later on.

For the *perfect* click model, the best online performance is achieved in the baseline setting for four out of nine data sets, ranging from 87.38 (*NP2004*, row 4) to 108.06 (*HP2003*, row 1). For these data sets, the exploitative baseline algorithm appears to learn well enough, so that additional exploration does not lead to high gains in offline performance that would outweigh its cost. For the three data sets *TD2003* (row 5), *TD-2004* (row 6), and *MQ2008* (row 9), online performance is higher at $\epsilon = 0.2$ than in the baseline setting, but the difference is not statistically significant. For the remaining data sets, we see statistically significant improvements over the baseline at $\epsilon = 0.4$. Online performance improves by 58% for the data set *OHSUMED*, and by 6% for the data set *MQ2007* (rows 7–8).

In the *navigational* click model, optimal online performance is achieved at higher exploration rates than for the *perfect* click model. For five data sets, the best setting is $\epsilon = 0.2$, and for four data sets it is $\epsilon = 0.4$. For all but one data set (*MQ2008*, row 18) the improvements in online performance over the baseline are statistically significant. Under noisier feedback, learning becomes more difficult, meaning that the quality of the learned weight vectors that can be exploited is lower than under perfect feedback. This reduces the benefit of exploitation, and lowers the cost of exploration, increasing the relative benefit of exploration. The biggest performance gains under increased exploration are observed for *NP2004* (row 13) and *TD2003* (row 14), where the online performance obtained in the best exploratory setting is 2.5 and 1.2 times that of the baseline setting. High performance gains are also observed for *HP2003* (81% improvement over the baseline, row 10), *HP2004* (98% improvement, row 11), *NP2003* (72.8%, row 12), and *TD2004* (81.97%, row 15). The improvement for *OHSUMED* is 51% (row 16). Small improvements are observed for *MQ2007* (6%, row 17) and *MQ2008* (1%, not statistically significant, row 18).

Compared to the *perfect* click model, online performance with the *navigational* model is much lower, as expected. The performance loss due to noise is between 2.7% (*NP-*

ϵ		**0.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
		perfect click model					
1	*HP2003*	**108.06**	99.96$^{\triangledown}$	87.58$^{\blacktriangledown}$	76.04$^{\blacktriangledown}$	50.75$^{\blacktriangledown}$	1.00$^{\blacktriangledown}$
2	*HP2004*	**101.75**	84.38$^{\blacktriangledown}$	73.88$^{\blacktriangledown}$	60.47$^{\blacktriangledown}$	42.22$^{\blacktriangledown}$	0.89$^{\blacktriangledown}$
3	*NP2003*	**104.51**	98.67	89.72$^{\blacktriangledown}$	72.00$^{\blacktriangledown}$	46.31$^{\blacktriangledown}$	1.57$^{\blacktriangledown}$
4	*NP2004*	**87.38**	84.33	75.76$^{\blacktriangledown}$	65.25$^{\blacktriangledown}$	44.62$^{\blacktriangledown}$	0.98$^{\blacktriangledown}$
5	*TD2003*	50.18	**50.54**	39.69$^{\blacktriangledown}$	28.55$^{\blacktriangledown}$	16.15$^{\blacktriangledown}$	1.94$^{\blacktriangledown}$
6	*TD2004*	47.49	**48.82**	34.00$^{\blacktriangledown}$	23.77$^{\blacktriangledown}$	13.57$^{\blacktriangledown}$	3.29$^{\blacktriangledown}$
7	*OHSUMED*	49.31	**78.07**$^{\blacktriangle}$	69.94$^{\blacktriangle}$	60.51$^{\blacktriangle}$	49.94	37.76$^{\blacktriangledown}$
8	*MQ2007*	64.59	67.56$^{\blacktriangle}$	**68.35**$^{\blacktriangle}$	63.99	57.88$^{\blacktriangledown}$	51.14$^{\blacktriangledown}$
9	*MQ2008*	89.20	**89.67**	85.74$^{\blacktriangledown}$	80.58$^{\blacktriangledown}$	73.78$^{\blacktriangledown}$	66.70$^{\blacktriangledown}$
		navigational click model					
10	*HP2003*	50.01	**90.34**$^{\blacktriangle}$	88.31$^{\blacktriangle}$	80.38$^{\blacktriangle}$	51.38	0.99$^{\blacktriangledown}$
11	*HP2004*	41.80	**82.76**$^{\blacktriangle}$	76.73$^{\blacktriangle}$	65.72$^{\blacktriangle}$	46.23	0.92$^{\blacktriangledown}$
12	*NP2003*	49.21	78.67$^{\blacktriangle}$	**85.03**$^{\blacktriangle}$	74.73$^{\blacktriangle}$	49.90	1.68$^{\blacktriangledown}$
13	*NP2004*	24.75	73.31$^{\blacktriangle}$	**86.53**$^{\blacktriangle}$	75.96$^{\blacktriangle}$	53.16$^{\blacktriangle}$	0.89$^{\blacktriangledown}$
14	*TD2003*	16.65	**36.53**$^{\blacktriangle}$	34.26$^{\blacktriangle}$	25.99$^{\blacktriangle}$	15.37	2.01$^{\blacktriangledown}$
15	*TD2004*	22.07	**40.16**$^{\blacktriangle}$	32.05$^{\blacktriangle}$	22.85	13.31$^{\blacktriangledown}$	3.30$^{\blacktriangledown}$
16	*OHSUMED*	46.16	**69.63**$^{\blacktriangle}$	66.28$^{\blacktriangle}$	58.58$^{\blacktriangle}$	48.77$^{\blacktriangle}$	37.83$^{\blacktriangledown}$
17	*MQ2007*	58.66	60.74$^{\blacktriangle}$	**62.08**$^{\blacktriangle}$	60.39$^{\blacktriangle}$	56.68$^{\blacktriangledown}$	51.21$^{\blacktriangledown}$
18	*MQ2008*	79.53	79.60	**80.38**	77.70$^{\triangledown}$	72.85$^{\blacktriangledown}$	66.23$^{\blacktriangledown}$
		informational click model					
19	*HP2003*	4.26	12.47$^{\blacktriangle}$	38.36$^{\blacktriangle}$	**46.37**$^{\blacktriangle}$	39.11$^{\blacktriangle}$	0.97$^{\blacktriangledown}$
20	*HP2004*	2.54	16.01$^{\blacktriangle}$	30.98$^{\blacktriangle}$	**39.85**$^{\blacktriangle}$	28.09$^{\blacktriangle}$	0.93$^{\blacktriangledown}$
21	*NP2003*	3.87	9.44$^{\blacktriangle}$	25.48$^{\blacktriangle}$	**41.97**$^{\blacktriangle}$	38.29$^{\blacktriangle}$	1.60$^{\blacktriangledown}$
22	*NP2004*	2.28	10.97$^{\blacktriangle}$	31.76$^{\blacktriangle}$	**49.12**$^{\blacktriangle}$	37.71$^{\blacktriangle}$	0.95$^{\blacktriangledown}$
23	*TD2003*	1.66	7.28$^{\blacktriangle}$	14.17$^{\blacktriangle}$	**16.03**$^{\blacktriangle}$	10.62$^{\blacktriangle}$	1.96$^{\vartriangle}$
24	*TD2004*	4.71	14.09$^{\blacktriangle}$	**20.03**$^{\blacktriangle}$	17.45$^{\blacktriangle}$	10.85$^{\blacktriangle}$	3.25$^{\blacktriangledown}$
25	*OHSUMED*	36.77	49.75$^{\blacktriangle}$	**59.85**$^{\blacktriangle}$	55.79$^{\blacktriangle}$	48.00$^{\blacktriangle}$	37.81
26	*MQ2007*	55.02	56.33$^{\vartriangle}$	56.42$^{\blacktriangle}$	**56.87**$^{\blacktriangle}$	55.06	51.14$^{\blacktriangledown}$
27	*MQ2008*	**72.68**	72.22	72.36	72.15	70.85$^{\blacktriangledown}$	66.33$^{\blacktriangledown}$

Table 6.1: Results for the pairwise approach. Online performance (in terms of cumulative NDCG) over 1,000 iterations for the exploitative baseline $\epsilon = 0$ and increasingly exploratory runs ($\epsilon > 0$).

2004) and 27.7% (*TD2003*), when comparing the best settings for each click model and data set.

In the noisier *informational* click model, the trends observed for the *navigational* click model continue. Performance in the purely exploitative setting is substantially lower than for the other click models, as the increase in noise makes learning more difficult. Compared to the *navigational* click model, online performance drops by another 8% (*MQ2007*) to 51% (*HP2004*).

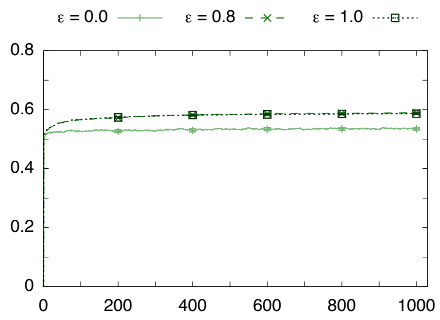
Under this click model, the cost of exploration further decreases relative to its benefit, so optimal performance is again seen at higher exploration rates. For six data sets, the best online performance is achieved at $\epsilon = 0.6$; for two data sets the best setting is $\epsilon = 0.4$. All improvements are statistically significant when compared to the purely exploitative baseline. For the *HP*, *NP*, and *TD* data sets, online performance improves by as much as an order of magnitude (rows 19–24). For the remaining data sets, improvements are lower, at 63% (*OHSUMED*, row 25) and 3% (*MQ2007*, row 26). An exception is *MQ2008*, for which there are no significant differences in online performance for runs with $\epsilon \in [0.0, 0.6]$ (row 27).

Overall, we conclude that balancing exploration and exploitation for the pairwise approach can lead to significant and substantial improvements in online performance. This balance appears to be strongly affected by noise, with highest relative improvements observed under the noisiest *informational* click model. Also, as click noise increases, the amount of exploration required for good online performance increases. Best values are $\epsilon \in [0.0, 0.2]$ for the *perfect* click model, $\epsilon \in [0.2, 0.4]$ for the *navigational* click model, and $\epsilon = [0.4, 0.6]$ for the *informational* click model. These findings confirm our hypothesis that balancing exploration and exploitation in the pairwise approach improves online performance.

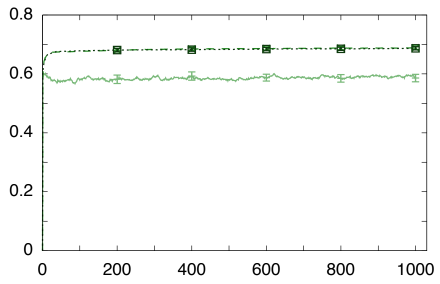
Besides overall trends in online performance under different exploration rates, we find performance differences between data sets. One such difference is that for the *HP* and *NP* data sets online performance tends to be higher than the remaining data sets, especially under the *perfect* and *navigational* click models. This suggests that these data sets are easier, i.e., that click feedback can be used effectively to learn linear weight vectors that generalize well. We can confirm this analysis by comparing the offline performance that the pairwise approach achieves on these data sets. For the *perfect* click model, an overview is included in Table 6.3 (on page 109). Indeed, offline NDCG@10 ranges from 0.704 (*HP2004*) to 0.760 (*HP2003*) for the "easy" data sets, and is substantially lower for the more difficult data sets (from 0.272 for *TD2003* to 486 for *MQ2008*). While our NDCG scores are not directly comparable with those reported by Liu (2009) (only NDCG@1 scores are equivalent, cf., §2.1), they show the same trend in terms of relative difficulty.

Under the *perfect* click model we found differences between most data sets and *OHSUMED* and *MQ2007*. For these two data sets, online performance increased significantly at $\epsilon = 0.2$, while for the remaining data sets, no significant improvements over the purely exploitative baseline were observed. The significant improvements at an increased exploration rate suggest that either big learning gains were realized for these data sets with increased exploration (outweighing the cost of exploraiton), or that exploration for these data sets is relatively low. As we detail below, both effects play a role here.

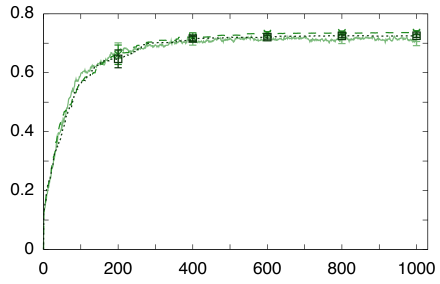
To analyze performance differences between the data sets, we study the learning curves of these data sets at different levels of exploration. Figure 6.1 shows the offline performance in terms of NDCG (on the whole result list) plotted over time (up to 1,000 iterations) for the data sets *MQ2007*, *OHSUMED*, *NP2003*, and *HP2004*. For the data sets *MQ2007* (Figure 6.1(a)) and *OHSUMED* (Figure 6.1(b)) we see that the best offline performance is achieved at high exploration rates (the dark, dashed and dotted lines; the difference between settings $\epsilon = 0.8$ and $\epsilon = 1.0$ is negligible). For *MQ2007*, offline performance at $\epsilon = 1.0$ is 0.533, 3% higher than in the baseline setting. The biggest



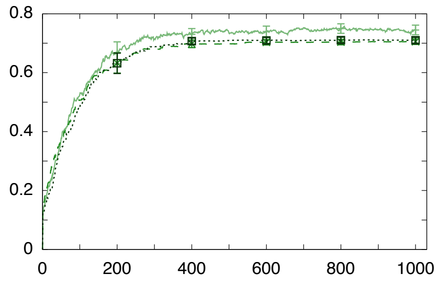
(a) *MQ2007*



(b) *OHSUMED*



(c) *NP2003*



(d) *HP2004*

Figure 6.1: Final performance for the pairwise approach over time for the data sets *MQ2007*, *OHSUMED*, *NP2003*, and *HP2004*, under the *perfect* click model and $\epsilon \in \{0.0, 0.8, 1.0\}$.

difference between the final performance of the exploitative baseline and higher levels of exploration under the *perfect* click model is observed for the data set *OHSUMED* (Figure 6.1(b), offline performance is 0.657 for $\epsilon = 1.0$, 9% higher than in the baseline setting). For this data set, the pairwise algorithm learns very poorly without at least some exploration. Not shown is the learning curve for *MQ2008*. It follows the same trend, with a final difference in offline performance of 4% (offline performance is 0.497 when $\epsilon = 1.0$).

Different behavior is observed for the remaining data sets. For the data sets *NP2003* (Figure 6.1(c)), *HP2003*, *TD2003*, and *TD2004* (not shown) there is no significant difference in offline performance between less and more exploratory settings under perfect feedback. This is contrary to the expected behavior that the highest level of exploration should result in best learning, as pure exploration corresponds to randomly sampling document pairs for preference detection. Most likely, this unexpected behavior under the *perfect* click model results from an effect similar to that observed in active learning. Because the current top results are shown, feedback is focused on the part of the document space that is most informative for learning. The data set for which this effect is observed has only few relevant documents, so that focusing feedback on a promising region can have a substantial benefit. The strongest effect is seen for data set *HP2004* (Figure 6.1(d)), where offline performance improves when implicit feedback is collected on exploitative result lists ($\epsilon = 0$, light and solid line) as opposed to more exploratory settings.

Besides the gains in offline performance realized under the *perfect* click model for *OHSUMED* and *MQ2007* under increased exploration rates, we can also confirm the relatively low risk of exploration for these data sets. In Table 6.1 we see that under pure exploration, the drop in online performance for these two data sets and *MQ2008* is much smaller than for the remaining data sets. For example, online performance for *MQ2008* at $\epsilon = 1.0$ is 66.7 (row 9), which corresponds to an NDCG of 0.3–0.4 for the average result list presented to the user during learning. In contrast, online performance at this level of exploration is 0.89 for data set *HP2004* (row 2), which corresponds to an average NDCG of less than 0.005. These differences are a result of the number of candidate documents per query, and the relative ratio of relevant to non-relevant documents provided per query. As described in §3.4, *OHSUMED* and the *MQ* data sets have much fewer candidate documents per query (approximately a factor of 10, compared to the other data sets), and a much higher ratio of relevant to non-relevant documents. Under these conditions, randomizing candidate documents has a much smaller negative effect on online performance than for data sets with many (non-relevant) candidate documents. This results in the low cost of exploration observed for these data sets. For the *HP*, *NP*, and *TD* data sets, the low ratio of relevant to non-relevant documents results in a much higher cost of exploration.

While the low number of candidate documents for *OHSUMED* and the *MQ* data sets results in a low cost of exploration, they also reduce its benefit. Comparing the learning curves in Figure 6.1(a)–6.1(b) to those in Figure 6.1(c)–6.1(d), we see that a much smaller gain in offline performance is realized (the increase in offline performance over, e.g., the first 100 iterations is much smaller). Thus, for data sets with a high ratio of relevant documents, exploration is cheap, but its benefit is limited. An extreme case is *MQ2008*, where the benefit of improving offline performance through increased exploration is so small that it does not lead to significant improvements in online performance (rows 9, 18, and 27 in Table 6.1). More generally, we find that the balance of exploration and exploitation is affected by the magnitude of the learning gains (in terms of offline performance) that can be realized under increased exploration, and the cost of the exploration.

For all data sets, the absolute difference in final performance at varying exploration rates is relatively small under the *perfect* click model. Much higher variance is observed when we simulate noisy feedback. Figure 6.2 shows learning curves for the data set *NP-2003* at different settings of ϵ for the *navigational* and *informational* click models. For the *navigational* click model (Figure 6.2(a)) final performance improves over time for all $\epsilon > 0.0$.

For the *informational* click model, final performance degrades dramatically in the purely exploitative baseline settings ($\epsilon = 0,\ 0.102$). In this setting, performance decreases over time. The purely exploratory setting ($\epsilon = 1.0$) leads to reasonable final performance, while the best performance is achieved with high exploration and some exploitation ($\epsilon = 0.8,\ 0.724$). This finding also confirms our earlier observation that

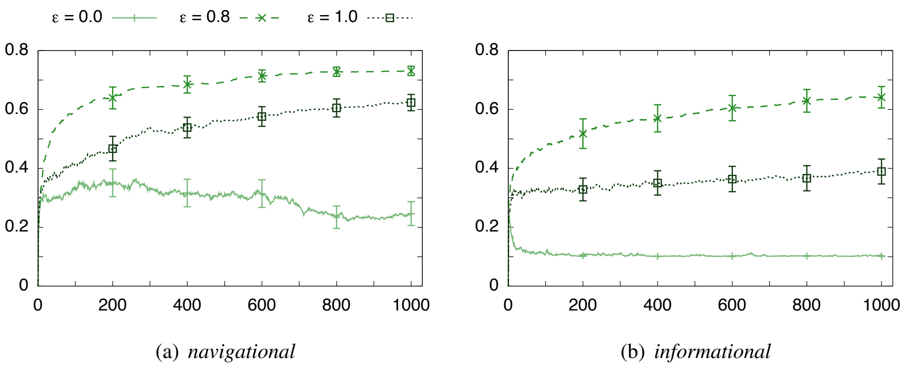


Figure 6.2: Final performance for the pairwise approach over time for the data set *NP-2003* for *navigational*, and *informational* click models and $\epsilon \in \{0.0, 0.8, 1.0\}$.

best offline performance is not always achieved in the most exploratory setting (possibly because feedback under increased exploitation focuses on promising documents).

Our analysis of offline performance results in a number of observations. We hypothesized that the best learning would occur with perfect feedback and pure exploration because this setting minimizes variance and bias in user feedback. As expected, learning outcomes were best for perfect feedback and degraded with noisier feedback. However, the effect of the exploration rate changed with the amount of noise in user feedback and characteristics of the data set. For perfect feedback, little to no exploration sometimes produced the best learning outcomes because exploitative result lists focused feedback on more informative parts of the solution space. For data sets with a low ratio of relevant to non-relevant documents, the low cost of exploration resulted in significant gains in online performance under reliable feedback. Under noisy feedback, higher exploration rates generally improved learning, though the best performance occurred with moderate amounts of exploitation. Overall, our results confirmed that balancing exploration and exploitation can significantly and substantially improve online performance in pairwise online learning to rank for IR.

6.3.2 Listwise Learning

Our main results for the listwise approach are shown in Table 6.2. The experiments described in §6.2.2 measure online performance of the exploratory baseline approach ($k = 0.5$) and increasingly exploitative ($k < 0.5$) experimental runs on the 9 LETOR 3.0 and 4.0 data sets on the *perfect*, *navigational*, and *informational* click models.

In the listwise setting, we expect best learning (in terms of offline performance) for the exploratory baseline approach. However, the online performance of the baseline approach is expected to be low, as it does not sufficiently exploit what has been learned. We hypothesize that increasing exploitation can improve online performance as long as its benefits outweigh the resulting loss in offline performance.

For the *perfect* click model, all data sets except *MQ2008* (row 9) improve over the

k		**0.5**	**0.4**	**0.3**	**0.2.**	**0.1**
				perfect click model		
1	*HP2003*	102.89	113.60$^\blacktriangle$	116.82$^\blacktriangle$	**122.38**$^\blacktriangle$	122.36$^\blacktriangle$
2	*HP2004*	95.81	103.38$^\blacktriangle$	108.87$^\blacktriangle$	**112.76**$^\blacktriangle$	109.71$^\blacktriangle$
3	*NP2003*	95.41	101.24$^\blacktriangle$	107.35$^\blacktriangle$	**110.24**$^\blacktriangle$	108.66$^\blacktriangle$
4	*NP2004*	99.67	108.41$^\blacktriangle$	114.83$^\blacktriangle$	**118.01**$^\blacktriangle$	117.87$^\blacktriangle$
5	*TD2003*	38.97	41.19$^\vartriangle$	43.86$^\blacktriangle$	**44.59**$^\blacktriangle$	42.72$^\blacktriangle$
6	*TD2004*	35.32	37.99$^\blacktriangle$	39.75$^\blacktriangle$	**42.01**$^\blacktriangle$	40.49$^\blacktriangle$
7	*OHSUMED*	69.03	71.78$^\blacktriangle$	74.47$^\blacktriangle$	**75.08**$^\blacktriangle$	75.02$^\blacktriangle$
8	*MQ2007*	59.66	61.50$^\blacktriangle$	61.81$^\blacktriangle$	**61.86**$^\blacktriangle$	61.86$^\blacktriangle$
9	*MQ2008*	77.90	78.05	**79.17**	78.98	77.86
				navigational click model		
10	*HP2003*	84.07	98.98$^\blacktriangle$	103.77$^\blacktriangle$	**108.43**$^\blacktriangle$	106.28$^\blacktriangle$
11	*HP2004*	73.83	85.14$^\blacktriangle$	88.74$^\blacktriangle$	91.22$^\blacktriangle$	**95.74**$^\blacktriangle$
12	*NP2003*	76.23	87.38$^\blacktriangle$	92.50$^\blacktriangle$	**96.94**$^\blacktriangle$	93.71$^\blacktriangle$
13	*NP2004*	83.75	95.93$^\blacktriangle$	97.89$^\blacktriangle$	106.28$^\blacktriangle$	**107.36**$^\blacktriangle$
14	*TD2003*	31.41	34.04$^\vartriangle$	35.39$^\blacktriangle$	37.26$^\blacktriangle$	**37.61**$^\blacktriangle$
15	*TD2004*	30.72	33.17$^\blacktriangle$	**34.62**$^\blacktriangle$	33.29$^\blacktriangle$	33.18$^\vartriangle$
16	*OHSUMED*	67.06	69.13$^\blacktriangle$	70.45$^\blacktriangle$	**71.72**$^\blacktriangle$	70.47$^\blacktriangle$
17	*MQ2007*	56.46	57.20	58.30$^\vartriangle$	**58.63**$^\blacktriangle$	57.73
18	*MQ2008*	74.84	74.70	**76.79**$^\vartriangle$	76.04	76.01
				informational click model		
19	*HP2003*	49.82	60.39$^\blacktriangle$	65.60$^\blacktriangle$	71.91$^\blacktriangle$	**75.68**$^\blacktriangle$
20	*HP2004*	44.76	48.39	55.69$^\blacktriangle$	**61.14**$^\blacktriangle$	60.41$^\blacktriangle$
21	*NP2003*	47.72	58.31$^\blacktriangle$	64.14$^\blacktriangle$	66.42$^\blacktriangle$	**77.17**$^\blacktriangle$
22	*NP2004*	48.64	63.44$^\blacktriangle$	66.43$^\blacktriangle$	**79.94**$^\blacktriangle$	78.74$^\blacktriangle$
23	*TD2003*	21.81	22.67	24.73$^\vartriangle$	**26.53**$^\blacktriangle$	25.83$^\blacktriangle$
24	*TD2004*	22.02	22.68	**24.50**$^\blacktriangle$	21.36	21.99
25	*OHSUMED*	62.83	63.47	**65.17**	63.81	61.02
26	*MQ2007*	54.89	54.79	**55.45**	54.66	55.12
27	*MQ2008*	71.38	72.43	72.77	71.93	**73.17**

Table 6.2: Results for the listwise approach. Online performance over 1,000 iterations for *baseline* ($k = 0.5$) and *exploit* ($k \in [0.1, 0.4]$) runs.

purely exploratory baseline for all settings of $k < 0.5$. For all these data sets, the best online performance is obtained at a relatively low setting of $k = 0.2$. Increases in online performance over the baseline range from 4% (*MQ2007*, row 8) to 19% (*HP2003*, row 1, and *TD2004*, row 6). The data set *MQ2008* is an exception. Although online performance is highest for $k = 0.3$, none of the exploitative settings perform significantly differently from the exploratory baseline. As discussed in the previous chapter, this data set has fewer candidate documents than other data sets, leading to a relatively low benefit of increased exploitation.

Results for the *navigational* click model are similar. For all data sets, online performance is significantly higher under higher exploitation than in the baseline setting. Best performance is achieved for $k \in [0.1, 0.3]$. For two data sets, $k = 0.3$ performs best. For four of the remaining data sets, best performance is achieved at $k = 0.2$, and for three data sets, best online performance is achieved at $k = 0.1$. Improvements of the best setting of k over the baseline range from 3% (*MQ2008*, row 18) to 30% (*HP2004*, row 11). As expected, performance under the *navigational* click model is lower than under *perfect* feedback.

The trend continues for the *informational* click model. Again, more exploitative settings of k outperform the purely exploratory baseline in all cases. For six out of nine cases, the improvements are statistically significant. These improvements range from 11% (*TD2004*, row 24) to 64% for the data set *NP2004* (row 22). For the remaining three data sets, no statistically significant differences between baseline and exploitative runs are observed, but small increases over the exploratory baseline are observed at smaller k.

Together, these results demonstrate that, for all click models and all data sets, balancing exploration and exploitation in listwise learning to rank for IR can significantly improve online performance over the purely exploratory baseline, which confirms our hypothesis. The best overall setting for the exploration rate is $k = 0.2$. This means that by injecting, on average, only two documents from an exploratory list, the algorithm learns effectively and achieves good online performance for all levels of noise. We conclude that the original listwise algorithm explores too much and surprisingly little exploration is sufficient for good performance.

Online performance is affected by noise in click feedback, as observed in the results obtained for the different click models. Performance is highest with *perfect* feedback, and decreases as feedback becomes noisier. Performance on some data sets is more strongly affected by noisy feedback. For the *HP*, *NP*, and *TD* data sets, performance for the *informational* model drops substantially. This may again be related to the large number of non-relevant documents in these data sets. Because finding a good ranking is harder, noise has a stronger effect. Despite this drop in performance, balancing exploration and exploitation consistently leads to better cumulative performance than the purely exploratory baseline for all levels of noise.

As for the pairwise approach, we analyze the relationship between online and offline performance by examining the learning curves for different levels of exploration. Figure 6.3 shows the learning curves for the data sets *MQ2007* and *NP2003* at different settings of k and the *perfect* click model. In contrast to the pairwise approach, there is no significant difference in performance after 1,000 iterations. We find the same behavior for all data sets. For *NP2003*, learning in the fully exploratory setting ($k = 0.5$) is slightly faster than in other settings. This is expected, as the best feedback is available at maximal exploration. However, learning at lower exploration rates quickly catches up. Thus, for the listwise approach, the exploration rate does not appear to have a significant effect on offline performance when feedback is perfect.

Learning curves for the *navigational* and *informational* click models for the data set *NP2003* are shown in Figure 6.4. As expected, learning is faster when feedback is more reliable. For the idealized *perfect* click model, offline performance after 1,000 iterations ranges between 0.719 ($k = 0.1$) and 0.727 ($k = 0.5$) for different settings of k. For

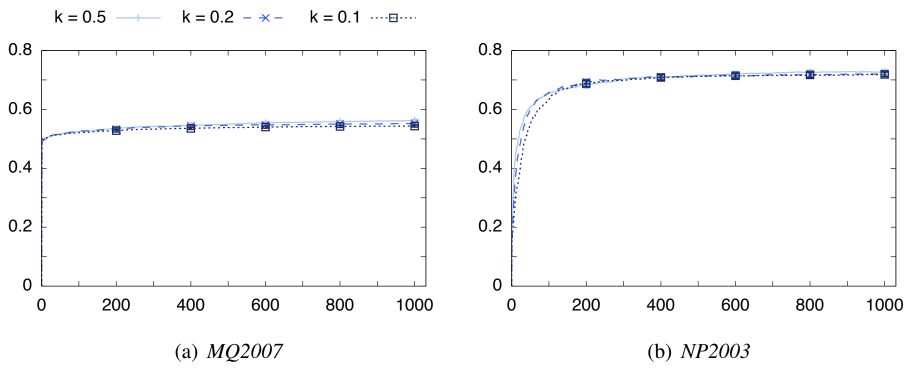


Figure 6.3: Offline performance (computed on the test set after each learning step) over time for the data sets *MQ2007* and *NP2003* for the *perfect* click model and $k \in \{0.1, 0.2, 0.5\}$.

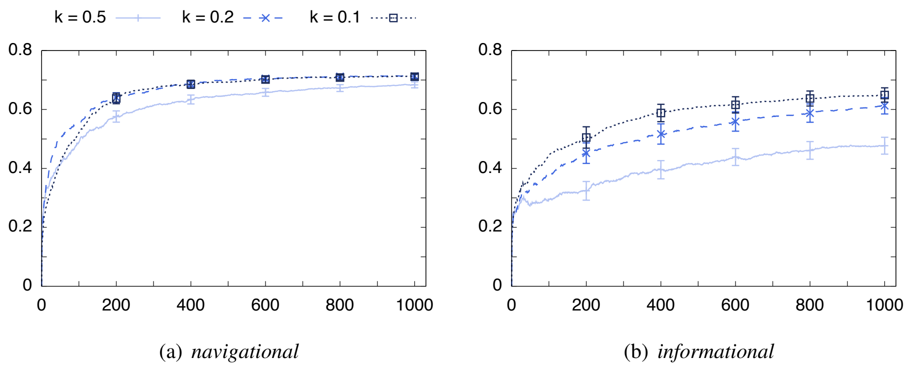


Figure 6.4: Final performance (with 5% confidence intervals) over time for the data set *NP2003* for *navigational*, and *informational* click models and $k \in \{0.1, 0.2, 0.5\}$.

the noisy *informational* click model, final performance is between 0.477 ($k = 0.5$) and 0.649 ($k = 0.5$). Although final performance drops substantially as implicit feedback becomes extremely noisy, performance improves over time for all data sets as there is still a signal to learn from, i.e., relevant documents are more likely to be clicked than non-relevant ones.

Once again there is an interaction effect between click model and exploration rate, although it is different from that observed under the pairwise approach. Here, there is no significant difference between the final performance at different settings of k under the *perfect* click model. Under the *navigational* click model, the effect of noise is small, and offline performance is similar to the *perfect* click model. However, in the *informational* click model, variance increases and there is a large difference between offline performance at different settings of k. This is a direct and expected consequence of the noise in inferred feedback. More surprising is that final performance improves for smaller k, since we expected feedback to be most reliable for the fully exploratory setting $k = 0.5$. Instead, it appears that, since bias is only partially compensated for (cf., §6.1), the bias that remains at lower values of k smoothes over some of the noise in the click model. At lower exploration rates, fewer results from the exploratory list are presented and it becomes harder for the exploratory list to win the comparison. Thus, instead of noisier updates, the algorithm makes fewer, more reliable updates that, on average, result in greater performance gains.

6.3.3 Comparing the Pairwise and Listwise Approach

For both the pairwise and the listwise approaches, our results show that a balance between exploration and exploitation is needed to optimize online performance. The mechanisms of how such a balance affects online performance, however, differ between the two learning approaches. Below, we first compare the online and offline performance of both approaches. Then, we discuss how exploration impacts the performance of both approaches, and conclude with implications for putting them in practice.

Table 6.3 gives an overview of the offline performance of the pairwise and listwise approaches in their best-performing setting under *perfect* click feedback. Like the online performance, these are computed after 1,000 iterations (consisting of one query, result list, and learning step each), which means that learning may not have converged and higher results are possible. These results should therefore be interpreted as a rough indication of what performance can typically be achieved by this approach in an online learning setting with relative feedback.

	pairwise			**listwise**		
	N@1	**N@3**	**N@10**	**N@1**	**N@3**	**N@10**
HP2003	**0.687**	0.727	0.760	0.684	**0.730**	**0.761**
HP2004	0.559	0.650	0.704	**0.582**$^{\triangle}$	**0.673**$^{\triangle}$	**0.725**$^{\triangle}$
NP2003	**0.531**	0.649	**0.705**	**0.531**	**0.650**	0.704
NP2004	**0.529**	**0.658**	**0.714**	0.521	0.656	0.710
TD2003	0.247	0.271	0.272	**0.318**$^{\triangle}$	**0.300**$^{\triangle}$	**0.295**$^{\triangle}$
TD2004	0.315	0.307	0.275	**0.385**$^{\triangle}$	**0.343**$^{\triangle}$	**0.300**$^{\triangle}$
OHSUMED	**0.515**	**0.474**	**0.444**	0.510	0.470	0.441
MQ2007	**0.352**	**0.359**	**0.400**	0.329$^{\triangledown}$	0.338$^{\triangledown}$	0.381$^{\triangledown}$
MQ2008	**0.347**	**0.390**	**0.486**	0.333$^{\triangledown}$	0.376$^{\triangledown}$	0.475$^{\triangledown}$

Table 6.3: Offline performance (in terms of NDCG@N) for the pairwise ($\epsilon = 1.0$) and listwise ($k = 0.5$) online learning to rank algorithms under the *perfect* click model.

In terms of offline performance, the pairwise and listwise approaches perform similarly. The pairwise approach outperforms the listwise approach on five out of nine data sets (in terms of NDCG@10), but the performance differences are significant for only two data sets (*MQ2007* and *MQ2008*). The listwise approach outperforms the pairwise approach on four data sets, in three cases significantly (*HP2004*, *TD2003*, and *TD2004*).

We note that the performance of the listwise approach is competitive, despite the limited information available to the algorithm (relative feedback per ranker instead of per document), and the weak information about the gradient that is inferred from this feedback (based on random exploration of the gradient instead of computing a gradient, as for the pairwise approach).

We compare the online performance of the pairwise and listwise approaches by comparing Tables 6.1 (page 101) and 6.2 (page 106). Under the *perfect* click model, and in the purely exploratory baseline setting, the listwise approach performs worse than the purely exploitative pairwise approach, as expected. However, at their optimal settings, the two approaches perform similarly, with the listwise approach beating the pairwise approach on four out of the nine data sets. We conclude that, under reliable feedback, the pairwise and listwise approaches perform similarly well when used with an appropriate balance of exploration and exploitation.

When click feedback is noisy, the listwise approach performs better than the pairwise approach. Under the *navigational* click model, the listwise approach outperforms the pairwise approach in terms of online performance on six data sets. Under the *informational* click model, this number increases to seven out of the nine data sets (at the optimal levels of exploration). The reason is that the approaches react to noise differently. For the pairwise approach in its exploitative baseline setting, increases in noise lead to dramatically reduced offline performance. However, balancing exploration and exploitation allows the algorithm to recover its performance. As a result, the optimal balance between exploration and exploitation shifts towards increased exploration as feedback becomes noisier. A relatively high amount of exploration, with about half the result list constructed from exploratory documents, is needed to achieve good learning outcomes. This relatively high amount of exploration, in turn, has a negative effect on online performance.

The drop in performance due to noise is much less pronounced for the listwise method. Online performance of the algorithm in its original, fully exploratory, version is often an order of magnitude higher than for the original version of the pairwise approach when feedback is noisy. A possible reason is that, by aggregating feedback over document lists, the algorithm becomes inherently robust to noise. Increasing exploitation can further improve online performance. While increases in exploitation introduce some amount of bias, this bias does not result in lower offline performance. Instead, it acts as a safeguard against too frequent updates based on noisy data. This leads to less frequent but more reliable updates of the weight vector, thereby improving offline performance. Thus, as noise in click feedback increases, a moderate level of exploitation can improve learning under the listwise approach.

Another advantage of the listwise approach is that the cost of exploration can be small if the exploratory document list is similar to the exploitative one, which is more likely as learning progresses. For the pairwise approach, the cost of exploration is generally high, so the approach has a disadvantage when a similar level of exploration is required for reasonable learning gains. Thus, at similar final performance and exploration rates, the listwise approach tends to achieve higher online performance than the pairwise approach.

Our analysis suggests that the pairwise and listwise approaches are appropriate for learning from relative feedback in different settings. When user feedback is reliable, the pairwise approach should be preferred as it results in good offline performance. Also, in this setting, the pairwise approach requires little to no exploration for good offline performance. It can exploit aggressively, leading to high online performance. However, when feedback is noisy, the listwise approach should be preferred. In contrast to the pairwise approach, it safeguards against dramatic loss in offline performance, as long as there is some signal in the feedback that prefers truly relevant documents. In addition, under noisy feedback, the listwise approach requires much less exploration than the pairwise approach, and the cost of exploration is lower.

6.4 Conclusion

In this chapter, we studied the effect of balancing exploration and exploitation on online learning to rank for IR. We introduced two methods for balancing exploration and exploitation in this setting, based on one pairwise and one listwise learning approach. To the best of our knowledge, these are the first algorithms that can achieve such a balance in a setting where only relative feedback is available.

Regarding the main research question addressed in this chapter, we found that balancing exploration and exploitation can substantially and significantly improve online performance in pairwise and listwise online learning to rank for IR. The effect of balancing exploration and exploitation is complex and there is an interaction between the amount of exploitation and the amount of noise in user feedback. When feedback is reliable, both pairwise and listwise approaches learn well and a high amount of exploitation can be tolerated, which leads to high online performance. As feedback becomes noisier, learning under high exploitation becomes unreliable for the pairwise approach. A higher amount of exploration is required to maintain reasonable performance. For the listwise approach, however, a smoothing effect occurs under high exploitation, so that learn well despite a high level of exploitation. This allows the listwise approach to maintain good performance under noisy feedback with a surprisingly small amount of exploration.

Our results also shed new light on the relative performance of online learning to rank methods. The pairwise approach makes effective use of implicit feedback when there is little noise, leading to high offline performance. However, it is strongly affected by noise in user feedback. Our results demonstrated that a balance of exploration and exploitation is crucial in such a setting, with more exploration needed as feedback becomes noisier. The offline performance of the listwise approach is similar to that of the pairwise approach under perfect feedback, but it is much more robust to noise, due to the aggregation of feedback over result lists. The listwise approach shows lower online performance than the pairwise approach in its purely exploratory baseline setting, but it performs well when exploration and exploitation are properly balanced. This first comparison of pairwise and listwise learning to rank in an online setting suggests that listwise approaches are a promising avenue of future development, because performance is competitive, robustness to noise is high, and only few approaches have been developed for the online setting (for learning with relative feedback).

The results of this chapter show that it is important to consider the effects of online learning to rank approaches on online performance. It is not sufficient to learn effectively, but by explicitly addressing online performance users can be provided with significantly better results throughout learning. We showed that balancing exploration and exploitation is one way in which online performance can be improved.

Our approach to balancing exploration and exploitation for listwise online learning to rank were based on a simple stochastic extension of BI. Bias introduced by increased exploitation could only be compensated for approximately, and had a complex effect of online and offline performance. A similar solution can be devised for TD. Using the probabilistic interleaving methods developed in Chapter 4, comparison outcomes can be inferred from a much larger family of distributions over result lists. This opens up a range of possibilities for constructing exploratory and exploitative result lists without introducing bias. The basic mechanisms of balancing exploration and exploitation, e.g., that increased exploitation at the same offline performance increases online performance, are expected to hold under all alternative approaches. However, more complex solutions, e.g., enabled by PI-MA-IS, are expected to lead to further gains in online performance in online learning to rank for IR.

In Chapter 4, we focused on methods for inferring accurate feedback through interleaved comparisons, but did not consider the effects of the developed evaluation approaches on online performance. In the next chapter (Chapter 7), we investigate how online performance is affected by existing interleaved comparison methods and the probabilistic approach developed in the earlier chapter. In particular, we investigate how to learn quickly and reliably from noisy user feedback in an online learning to rank setting.

7 Reusing Historical Interaction Data for Faster Learning

In our final research chapter, we investigate whether and how historical interaction data can be reused to speed up online learning to rank for IR. This chapter builds on the results of Chapter 4, in particular our interleaved comparison method for historical data reuse, PI-MA-IS.

Learning quickly from the limited quality and quantity of feedback that can be inferred from user interaction is a main challenge in learning to rank for IR. Learning speed is particularly important in terms of the number of user interactions. The better the system's performance is after a smaller number of interactions, the more likely users are to be satisfied with the system. Also, the more effective an online learning to rank algorithm is, the more feasible it is to adapt to smaller groups of users, or even individual users. Furthermore, user feedback is limited because the learning algorithm should be invisible to system users, i.e., feedback is inferred from natural (noisy) user interactions.

A limitation of current online learning to rank approaches for IR is that they utilize each observed data sample (consisting of a query, the displayed results, and observed user clicks on the result list) only once. This was necessary because it was not clear how feedback from previous user interactions (that were collected with different rankers) could be reused. The interleaved comparison method PI-MA-IS that we developed in Chapter 4 allows data reuse for ranker evaluation. It was found to be effective for making ranker comparisons more reliable, especially when large amounts of historical data were available. In this chapter, we investigate whether and how this evaluation method can be integrated with online learning to rank approaches, and whether and in what way these additional (historical, and possibly noisier or biased) evaluations can lead to faster learning.

The central research question addressed in this chapter is:

RQ 14 Can previously observed (historical) interaction data be used to speed up online learning to rank?

To answer this question, we develop the first two learning approaches for reusing historical data in online learning to rank for IR: *reliable historical comparisons* (RHC), which uses historical data directly to make feedback more reliable, and *candidate pre-selection* (CPS), which uses historical data to preselect candidate rankers. In extensive experiments, we investigate whether and how historical data reuse can speed up online learning to rank and lead to higher online performance. In addition, we analyze our results to answer the following more detailed questions:

RQ 15 Is historical data more effective when used to make comparisons more reliable (as in RHC), or when used to increase local exploration (as in CPS)?

RQ 16 How does noise in user feedback affect the reuse of historical interaction data for online learning to rank?

We find that historical data can be effectively reused to speed up online learning to rank for IR. Particularly effective is the CPS approach, which reuses historical data to preselect candidate rankers, and can thereby compensate for noise in user feedback. Our results directly impact the effectiveness of online learning to rank approaches, especially in settings where feedback may be noisy.

The remainder of this chapter is organized as follows. We detail our two approaches for reusing historical data in online learning to rank for IR in §7.1. We present our experiments and results in §7.2 and §7.3, and provide further analysis in §7.4. We conclude in §7.5.

7.1 Method

In this section, we detail our two approaches for online learning to rank for IR with reuse of historical data. Both are based on our problem formulation of online learning to rank as a contextual bandit problem (§3.1). Our methods are based on the listwise learning algorithm DBGD (§2.5.2), and on our probabilistic interleave methods PI-MA and PI-MA-IS (Chapter 4). Below, we detail our RHC method for reusing historical data for reliable comparisons (§7.1.1) and our CPS method for candidate preselection (§7.1.2).

7.1.1 Reliable Historical Comparison

Our first method is based on the idea of using historical interaction data to supplement live comparisons. This can improve the quality of interleaved comparisons in two ways. When a live comparison resulted in a tie (e.g., because no clicks were observed, or all clicks were on documents that were placed at the same ranks by both rankers), a ranker difference may still be detected on the historical interactions. In cases where live comparisons are noisy, they can be compared with comparisons on historical interaction data to improve reliability. The main challenge of such an approach is how to properly combine live and historical estimates. Here, we present an approach that combines estimates obtained using PI-MA and PI-MA-IS. We call this approach *reliable historical comparison* (RHC).

We define RHC as an extension to a listwise linear learner, such as DBGD (see Algorithm 5 on page 30) that uses PI-MA for ranker comparisons (Algorithm 7 on page 51). To enable historical data reuse in DBDG, we set $\lambda > 0$ (Algorithm 5, lines 1 and 11–14). Then, we use the collected historical data $\mathbf{h}$ to supplement the interleaved comparisons based on live data as shown in Algorithm 12. This algorithm replaces line 13 (computing live comparison outcomes) in Algorithm 7.

Algorithm 12 (RHC) Probabilistic interleaved comparison with reuse of historical data for use in DBDG (Algorithm 5). This algorithm computes combined comparison outcomes, e.g., as a replacement of line 13 in Algorithm 7.

```
1: Input: $o_L(\mathbf{l}, \mathbf{a}, \mathbf{c})$, $o_H(\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}', \mathbf{a}', \mathbf{c}')$, $\mathbf{l}_1$, $\mathbf{l}_2$, $\mathbf{l}$, $\mathbf{a}$, $\mathbf{c}$, $\mathbf{h} = n \times (\mathbf{l}'_i, \mathbf{a}'_i, \mathbf{c}'_i)$
2: $o_L \leftarrow o_L(\mathbf{l}, \mathbf{a}, \mathbf{c})$            // compute live outcome following Eq. 4.5
   // compute historical outcome, biased (Eq. 4.5) or unbiased (Eq. 4.10)
3: $\mathbf{o}_H \leftarrow []$
4: for $(\mathbf{l}'_i, \mathbf{a}'_i, \mathbf{c}'_i) \in \mathbf{h}$ do
5:     $append(\mathbf{o}_H, o_H(\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}', \mathbf{a}', \mathbf{c}'))$
6: $\beta_L \leftarrow 1$
7: $\beta_H \leftarrow var(\mathbf{o}_H)$
8: $o_C \leftarrow (\beta_L * mean(\mathbf{o}_H) + \beta_H * mean(o_L))/(\beta_L + \beta_H)$
9: return $o_C$
```

RHC takes as input two functions for computing outcomes: $o_L(\cdot)$, which accepts data from one live observation, and $o_H(\cdot)$, which accepts as input the current target lists as well as one historical observation. Furthermore, RHC takes as input the target lists $\mathbf{l}_1$ and $\mathbf{l}_2$ to be compared, one live observation, i.e., the interleaved list $\mathbf{l}$, assignments $\mathbf{a}$, and clicks $\mathbf{c}$ observed on the interleaved list. In addition, it accepts n historical data points that were observed in previous comparisons of other, original, result lists $\mathbf{l}'_1$ and $\mathbf{l}'_2$. The algorithm first generates the live outcome o_L as in the live setting, using the live outcome method $o_L(\cdot)$ (line 2). Then, additional outcome estimates $\mathbf{o}_H$ are computed using the historical data and $o_H(\cdot)$ (lines 4–5).

In this chapter, we instantiate $o_H(\cdot)$ in two ways to explore the effects of bias and variance on this approach. In Chapter 4, we showed that applying PI-MA directly to historical data results in biased estimates of comparison outcomes. The alternative method PI-MA-IS, which compensates for bias using importance sampling, is unbiased but can suffer from high variance when only little data is available. In the online evaluation setting we investigated in Chapter 4, we found that both methods are similarly effective for relatively small amounts of data, while for large amounts of data the unbiased method is more reliable. In contrast, in the online learning to rank task addressed here, we expect the effect of bias and variance to be relatively small, because the amounts of historical data are small, and because subsequent ranker pairs are more similar to those used to obtain the historical samples than in the evaluation setting addressed previously.

Our first (biased) instantiation of $o_H(\cdot)$ uses PI-MA to estimate outcomes for the target lists $\mathbf{l}_1$ and $\mathbf{l}_2$ given historical data (Eq. 4.5). It uses the historical $\mathbf{l}'$ and $\mathbf{c}'$ to compute comparison outcomes (and marginalizes over all possible assignments $\mathbf{a} \in \mathbf{A}$), but uses the target distribution P_T (based on the current target lists) to compute $P(\mathbf{a}|\mathbf{l}_i, q_i)$ (Eqs. 4.6–4.8).

The resulting comparison method computes outcomes based on historical data but may be biased. Under the current target rankers, document distributions may be different from those under which the historical data was collected. This means that it is not guaranteed that each target ranker has an equal chance of contributing its highly ranked documents to the interleaved list, and to obtain clicks on these documents. As a result, the target list that is more similar to the historical lists has an advantage over the less similar one.

Our second instantiation of $o_H(\cdot)$ uses PI-MA-IS to compensate for bias using importance sampling (Eq. 4.10). As in the biased scoring method, it uses the historical $\mathbf{l}'$ and $\mathbf{c}'$ to compute comparison outcomes. In contrast to the biased method, each outcome is then weighted by its probability of occurring under the target distribution (P_T) versus the original (historical) distribution (P_S). Intuitively, this means that observations that are more likely under the target distribution, and less likely under the original distribution, obtain a high weight and vice versa. As shown in Theorem 4.2.2, PI-MA-IS produces unbiased estimates of the comparison outcomes under the target distribution from historical data (collected under the source distribution). While this approach is unbiased, it may suffer from high variance when the target and source distributions are very different from each other, which may lead to unreliable outcome estimates.

After computing the live and historical estimates o_L and $\mathbf{o}_H$, they are combined into a final estimate o_C using the Graybill-Deal estimator (Graybill and Deal, 1959) (line 6–8). This combined estimator weights the two estimates by the ratio of their sample variances. It was shown to result in a minimal variance combined estimate when the variances of the individual estimators are known, and to have strictly lower variance than either individual estimate when their variances are estimated on samples of size $n > 10$ (Graybill and Deal, 1959). Here, the true variance of the estimators are unknown. For the historical estimator, we can use the sample variance as an estimate of the true variance (line 7).

A limitation of combining historical and live estimates according to Algorithm 12 is that for any given comparison we only have one live data point collected under the current target rankers, so that the variance of the live outcome(s) cannot be estimated. Here, we set the weight of the live outcomes to $\beta_L = 1$ (line 6).[1] Our experiments in §7.2 investigate whether this approximation is sufficient for improved performance. We hypothesize that the reliability of comparisons can be improved using RHC, leading to faster learning.

7.1.2 Candidate Preselection

Our second approach for reusing historical data to speed up online learning to rank for IR uses historical data to improve candidate generation. Instead of randomly generating a candidate ranker to test in each comparison, it generates a pool of candidate rankers and selects the most promising one using historical data. We hypothesize that historical data can be used to identify promising rankers and that the increased quality of candidate rankers can speed up learning. We call this second approach *candidate preselection* (CPS).

Like RHC, CPS is designed as an extension to DBGD (Algorithm 5). Again, we set $\lambda > 0$ to collect historical data. However, CPS uses the collected historical data, not during the comparison step, but for selecting candidate rankers.

Our implementation of CPS is shown in Algorithm 13, which replaces the method $generate\_candidate(\cdot)$ in DBGD. As input, it takes a comparison function $o_H(\cdot)$ that

[1]We also experimented with batches of comparisons where the same original pair was used for several subsequent comparisons. However, the performance loss due to the resulting smaller number of updates outweighed the gain due to improved variance estimates.

Algorithm 13 (CPS) Generating candidate rankers with preselection, for use as $g(\delta, \mathbf{w}_t)$ in Algorithm 5.

```
1: **Input**: $o_H(\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}', \mathbf{a}', \mathbf{c}')$, $\mathbf{w}_t$, $\delta$, $\eta$, $\zeta$, $\mathbf{h} = n \times (\mathbf{l}'_i, \mathbf{a}'_i, \mathbf{c}'_i)$
2: $\mathbf{e} = []$
   // generate candidate pool
3: for i in $(i = 1..\eta)$ do
4:     $append(\mathbf{e}, generate\_candidate(\delta, \mathbf{w}_t))$
   // compare and eliminate candidates using historical data
5: while $len(\mathbf{e}) > 1$ do
6:     $\mathbf{p} \leftarrow sample(\mathbf{e}, 2)$
7:     $o_H \leftarrow []$
8:     for $i$ $(i = 1..\zeta)$ do
9:         $(\mathbf{l}'_i, \mathbf{a}'_i, \mathbf{c}'_i) \leftarrow sample(\mathbf{h}, 1)$
10:        $append(o_H, o_H(l(\mathbf{p}[1].w), l(\mathbf{p}[2].w), \mathbf{l}', \mathbf{a}', \mathbf{c}'))$
11:    if $mean(o_H) < 0$ then
12:        $remove(\mathbf{e}, \mathbf{p}[2])$
13:    else if $mean(o_H) > 0$ then
14:        $remove(\mathbf{e}, \mathbf{p}[1])$
15:    else
16:        $remove(\mathbf{e}, sample(\mathbf{p}, 1))$
17: return $\mathbf{e}[0]$                                  // return remaining candidate
```

estimates comparison outcomes using historical data, a current weight vector $\mathbf{w}_t$, the step size δ, arguments η and ζ that determine the size of candidate pools and the number of historical comparisons to conduct per ranker pair, and a vector of historical observations $\mathbf{h}$.

The algorithm is called when a new candidate ranker is requested. It first generates a pool of η candidate rankers by calling the original $generate\_candidate(\cdot)$ function (Algorithm 6) (lines 3–4). The most promising ranker is determined in rounds, where in each round a randomly selected pair of rankers (line 6) competes. For each pair, ζ comparisons are performed on historical data points randomly sampled from $\mathbf{h}$ with replacement (8–10). After the individual historical estimates are obtained, their mean is used to determine which ranker to eliminate from the pool. If there is a winner (i.e., $o_H \neq 0$), the losing ranker is removed. Otherwise, one of the rankers is selected to be removed at random. When only one element remains in the candidate pool, it is returned as the most promising candidate.

Our candidate selection approach ensures that a single candidate is selected after a finite $((\eta - 1) \times \zeta)$ number of comparisons. Because only the best candidate needs to be selected, the randomized approach is expected to provide a good balance of effectiveness and efficiency. In cases where the compared candidates perform equally well, only one candidate needs to be retained (chosen randomly).

Like in §7.1.1 above, we investigate the effect of bias and variance on this approach by implementing the comparison method $o_H(\cdot)$ in two different ways. First, we instantiate $o_H(\cdot)$ as PI-MA (Eq. 4.5), which has low variance but may result in biased estimates of comparison outcomes under historical data. Second, we instantiate $o_H(\cdot)$ as PI-MA-IS (Eq. 4.10), which removes bias using importance sampling, but may be affected by variance more strongly.

We hypothesize that CPS can substantially improve the quality of candidate rankers available for online learning, leading to faster learning than using live data only. Regarding the biased and unbiased version of CPS, we expect only small performance differences, as the amount of historical data reused per live comparison is small.

7.2 Experiments

Our experiments are designed to investigate whether online learning to rank for IR can be sped up by using historical data. We use the same experimental setup as in Chapter 6, detailed in Chapter 3. Again, we run our experiments on the 9 LETOR 3.0 and 4.0 data sets. Click data is generated using the *perfect*, *navigational*, *informational*, and *almost random* click models as shown in Table 3.1. Online performance is measured in terms of discounted cumulative NDCG with discount factor $\gamma = 0.995$.

We compare the following three baseline runs and four experimental runs:

BI Baseline – learning with live data only, using BI (Joachims, 2003; Radlinski et al., 2008b) for interleaved comparisons as detailed in Algorithm 1 on page 20 (Chapter 2).

TD Baseline – learning with live data only, using TD (Radlinski and Craswell, 2010; Radlinski et al., 2008b) for interleaved comparisons as detailed in Algorithm 2 on page 21 (Chapter 2).

PI Baseline – learning with live data only, using PI-MA for interleaved comparisons as detailed in Algorithm 7 (with Eq. 4.5) on page 51 (Chapter 4).

RHC-B Uses historical data to infer more reliable feedback (Algorithm 12, §7.1.1), with *biased* comparison outcome estimates (PI-MA, Eq. 4.5).

RHC-U Uses historical data to infer more reliable feedback (Algorithm 12, §7.1.1), with *unbiased* comparison outcome estimates (PI-MA-IS, Eq. 4.10).

CPS-B Uses historical data for candidate preselection (Algorithm 13, §7.1.2) with *biased* comparison outcome estimates (PI-MA, Eq. 4.5).

CPS-U Uses historical data for candidate preselection (Algorithm 13, §7.1.2) with *unbiased* comparison outcome estimates (PI-MA-IS, Eq. 4.10).

For each data set, we run experiments over 1000 iterations (i.e., simulated interactions), and repeat each experiment 25 times and average results over all folds and repetitions. Parameters for the DBGD learning algorithm are selected to match those found to work best in previous work ($\mathbf{w}_0$ is initialized to zero, $\alpha = 0.01$, $\delta = 1$, cf., (Yue and Joachims, 2009)). For the remaining parameters, we report results on one setting (for CPS, $\eta = 6$, $\zeta = 10$, and $\lambda = 10$; for RHC, $\lambda = 10$). The sensitivity to specific parameter settings is analyzed in §7.4.

7.3 Results

In this section, we present the results of the experiments described in §7.2, to answer the main research question of this chapter: *Can historical interaction data be reused to speed online learning to rank?* In addition to our main question, we also investigate *how* historical data can best be reused (i.e., to improve the reliability of evaluations, as in our RHC approach, or to improve the quality of candidate rankers, as in CPS), and whether and how historical data reuse is affected by bias and variance in outcomes estimated from historical data.

Table 7.1 shows the online performance obtained on all LETOR 3.0 and 4.0 data sets, for the four click model instantiations specified in Table 3.1, the three baseline runs BI, TD, PI, and the four experimental runs RHC-B, RHC-U, CPS-B and CPS-U. For reasons discussed below, TD outperforms the other baseline methods. Therefore, we use TD as our baseline for significance testing.

Overall, we see that the highest online performance is achieved by our CPS method for all data sets and click models. The observed improvements over TD are statistically significant and substantial. For example, for the data set *HP2003*, the highest performance under the *perfect* click model is 116.85 (using biased estimates of comparison outcomes), which constitutes an improvement of 7.3% over the best-performing baseline method TD (cf., row 1). There are only four cases in which the observed improvements are not statistically significant: for *OHSUMED* and *MQ2008* under the *informational* click model (rows 25 and 27), and for *MQ2007* and *MQ2008* under the *almost random* click model (rows 35 and 36). In these cases, small improvements over the baseline are observed, but they are not statistically significant.

To put the obtained absolute online performance scores in perspective, recall that we measure discounted cumulative reward, i.e., high online performance is obtained when a method both learns well (i.e., it achieves high offline performance, in terms of NDCG), and it learns quickly, i.e., after a small number of interactions. In our experimental setup, a method that presented perfect result lists (with NDCG $= 1$) on all interactions could obtain an online performance of 200.0, while a method that would obtain no reward on the first 500 interactions and perfect results after would achieve an online performance of only 15.0. Scores obtained by our methods fall between these two extremes, indicating that good rankers are learned within a few hundred simulated interactions.

For the baseline methods, which learn from live data only, we find that online performance of BI and TD is very similar, while that of PI is significantly lower for most data sets and click models. To understand why, we compare the offline performance and learning speed of these methods (cf., the offline performance on data set *NP2003* in Figure 7.1).[2] We see that the final offline performance is very similar (differences are not statistically significant), and that they also learn equally fast. Thus, PI learns as well as BI and TD but loses online performance due to the increase in randomization during interleaving. Compared to the evaluation setting, where PI was shown to outperform BI and TD when applied to compare rankers over large amounts of data, PI performs worse (Chapter 4). PI provides fine-grained information about the magnitude of ranker

[2]We present offline performance plots for only one data set, because plots for the remaining data sets are qualitatively similar.

		BI	**TD**	**PI**	**RHC-B**	**RHC-U**	**CPS-B**	**CPS-U**
				perfect click model				
1	*HP2003*	107.84	108.94	97.62$^{\blacktriangledown}$	89.43$^{\blacktriangledown}$	96.54$^{\blacktriangledown}$	**116.85**$^{\blacktriangle}$	114.60$^{\blacktriangle}$
2	*HP2004*	99.82	99.72	89.72$^{\blacktriangledown}$	81.75$^{\blacktriangledown}$	88.25$^{\blacktriangledown}$	**108.87**$^{\blacktriangle}$	107.44$^{\blacktriangle}$
3	*NP2003*	97.94	98.15	88.67$^{\blacktriangledown}$	81.93$^{\blacktriangledown}$	87.36$^{\blacktriangledown}$	**108.50**$^{\blacktriangle}$	105.25$^{\blacktriangle}$
4	*NP2004*	102.96	102.72	93.50$^{\blacktriangledown}$	88.25$^{\blacktriangledown}$	93.04$^{\blacktriangledown}$	**114.95**$^{\blacktriangle}$	112.41$^{\blacktriangle}$
5	*TD2003*	40.38	38.81	36.21$^{\blacktriangledown}$	29.84$^{\blacktriangledown}$	33.34$^{\blacktriangledown}$	**46.93**$^{\blacktriangle}$	45.31$^{\blacktriangle}$
6	*TD2004*	36.19	35.87	34.55$^{\triangledown}$	27.62$^{\blacktriangledown}$	30.65$^{\blacktriangledown}$	**44.71**$^{\blacktriangle}$	43.04$^{\blacktriangle}$
7	*OHSUMED*	70.68	70.14	68.29$^{\blacktriangledown}$	61.51$^{\blacktriangledown}$	63.86$^{\blacktriangledown}$	**75.11**$^{\blacktriangle}$	74.80$^{\blacktriangle}$
8	*MQ2007*	59.87	60.18	58.23$^{\blacktriangledown}$	56.09$^{\blacktriangledown}$	57.96$^{\blacktriangledown}$	63.12$^{\blacktriangle}$	**63.73**$^{\blacktriangle}$
9	*MQ2008*	79.03	77.98	75.57$^{\blacktriangledown}$	76.67	77.28	**84.18**$^{\blacktriangle}$	83.44$^{\blacktriangle}$
				navigational click model				
10	*HP2003*	83.84	85.90	76.61$^{\blacktriangledown}$	78.05$^{\blacktriangledown}$	83.80	**112.20**$^{\blacktriangle}$	110.42$^{\blacktriangle}$
11	*HP2004*	75.32$^{\triangledown}$	80.02	68.91$^{\blacktriangledown}$	64.00$^{\blacktriangledown}$	73.45$^{\blacktriangledown}$	**104.14**$^{\blacktriangle}$	99.92$^{\blacktriangle}$
12	*NP2003*	77.90	79.99	72.06$^{\blacktriangledown}$	72.36$^{\blacktriangledown}$	77.22	**105.81**$^{\blacktriangle}$	102.97$^{\blacktriangle}$
13	*NP2004*	83.18$^{\triangledown}$	86.79	75.25$^{\blacktriangledown}$	78.51$^{\blacktriangledown}$	84.37	**111.09**$^{\blacktriangle}$	108.22$^{\blacktriangle}$
14	*TD2003*	31.74$^{\triangledown}$	33.92	30.41$^{\blacktriangledown}$	27.01$^{\blacktriangledown}$	29.60$^{\blacktriangledown}$	**43.63**$^{\blacktriangle}$	42.09$^{\blacktriangle}$
15	*TD2004*	31.12	31.05	29.19$^{\blacktriangledown}$	24.98$^{\blacktriangledown}$	27.72$^{\blacktriangledown}$	**40.38**$^{\blacktriangle}$	39.20$^{\blacktriangle}$
16	*OHSUMED*	67.50	67.64	65.18$^{\blacktriangledown}$	62.24$^{\blacktriangledown}$	61.79$^{\blacktriangledown}$	71.33$^{\blacktriangle}$	**71.76**$^{\blacktriangle}$
17	*MQ2007*	56.56	57.06	55.78$^{\triangledown}$	55.33$^{\blacktriangledown}$	55.40$^{\triangledown}$	59.60$^{\blacktriangle}$	**59.98**$^{\blacktriangle}$
18	*MQ2008*	74.14	74.51	72.78$^{\triangledown}$	74.45	73.11	**80.46**$^{\blacktriangle}$	79.12$^{\blacktriangle}$
				informational click model				
19	*HP2003*	55.63	55.65	46.87$^{\blacktriangledown}$	42.25$^{\blacktriangledown}$	64.04$^{\blacktriangle}$	**104.87**$^{\blacktriangle}$	100.16$^{\blacktriangle}$
20	*HP2004*	42.99	45.02	37.52$^{\blacktriangledown}$	35.91$^{\blacktriangledown}$	55.60$^{\blacktriangle}$	**92.05**$^{\blacktriangle}$	81.10$^{\blacktriangle}$
21	*NP2003*	53.38	52.88	45.38$^{\blacktriangledown}$	43.80$^{\blacktriangledown}$	63.59$^{\blacktriangle}$	**101.83**$^{\blacktriangle}$	98.64$^{\blacktriangle}$
22	*NP2004*	58.31	57.62	52.32$^{\triangledown}$	47.80$^{\blacktriangledown}$	72.70$^{\blacktriangle}$	**105.46**$^{\blacktriangle}$	98.18$^{\blacktriangle}$
23	*TD2003*	22.11	21.99	21.74	19.46$^{\triangledown}$	24.53$^{\vartriangle}$	**39.43**$^{\blacktriangle}$	37.00$^{\blacktriangle}$
24	*TD2004*	23.66	22.87	21.60	20.87$^{\triangledown}$	21.74	**28.49**$^{\blacktriangle}$	27.25$^{\blacktriangle}$
25	*OHSUMED*	63.39	64.91	60.48$^{\blacktriangledown}$	59.80$^{\blacktriangledown}$	57.06$^{\blacktriangledown}$	63.27	**65.39**
26	*MQ2007*	55.29	54.58	53.99	52.20$^{\blacktriangledown}$	54.77	56.41	**56.82**$^{\vartriangle}$
27	*MQ2008*	73.14	71.83	70.42	70.99	70.38	**74.56**	73.12
				almost random click model				
28	*HP2003*	38.27	40.90	36.32$^{\triangledown}$	49.58$^{\blacktriangle}$	58.02$^{\blacktriangle}$	**96.73**$^{\blacktriangle}$	89.29$^{\blacktriangle}$
29	*HP2004*	35.35	35.60	33.27	41.38$^{\vartriangle}$	49.76$^{\blacktriangle}$	79.82$^{\blacktriangle}$	**80.10**$^{\blacktriangle}$
30	*NP2003*	39.71	37.07	37.26	47.84$^{\blacktriangle}$	60.01$^{\blacktriangle}$	**93.94**$^{\blacktriangle}$	92.22$^{\blacktriangle}$
31	*NP2004*	40.20$^{\triangledown}$	45.57	41.33	54.67$^{\blacktriangle}$	65.20$^{\blacktriangle}$	90.78$^{\blacktriangle}$	**92.72**$^{\blacktriangle}$
32	*TD2003*	17.41	18.81	19.46	21.53$^{\blacktriangle}$	24.96$^{\blacktriangle}$	**35.36**$^{\blacktriangle}$	34.02$^{\blacktriangle}$
33	*TD2004*	18.92	19.23	18.90	20.14	19.63	**27.92**$^{\blacktriangle}$	25.42$^{\blacktriangle}$
34	*OHSUMED*	56.41	56.43	57.14	58.35	53.80$^{\triangledown}$	55.52	**61.78**$^{\blacktriangle}$
35	*MQ2007*	52.54$^{\triangledown}$	53.96	52.27$^{\triangledown}$	52.77	52.31$^{\triangledown}$	**54.70**	54.52
36	*MQ2008*	69.36	69.75	69.02	70.29	69.15	71.02	**71.47**

Table 7.1: Online performance (in terms of discounted cumulative reward) when learning with interleaved comparison methods. Statistical significance is indicated in comparison with the baseline method TD.

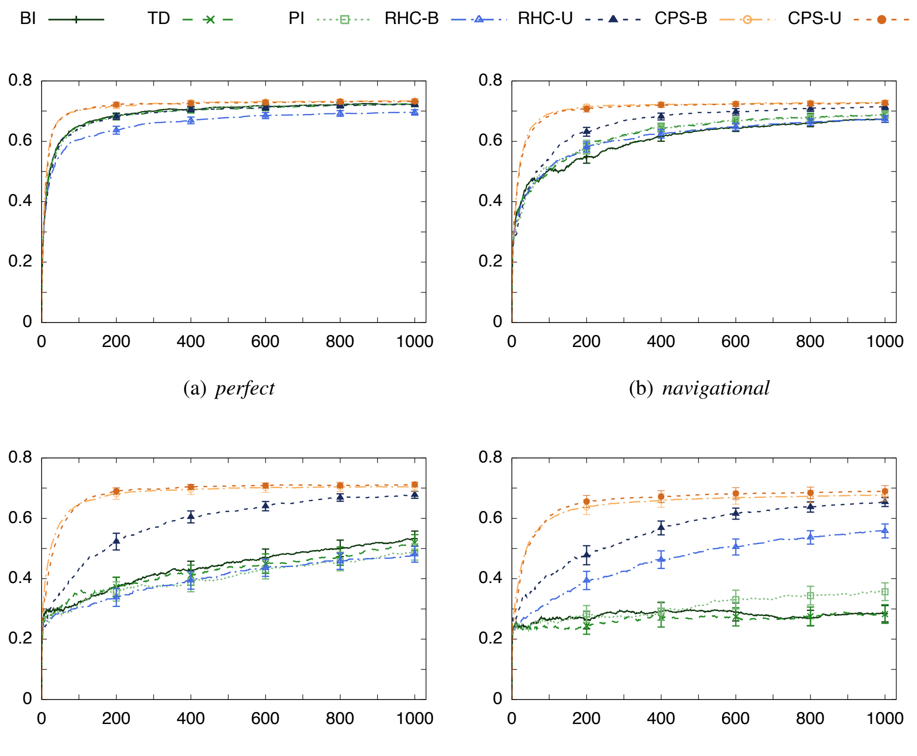
differences, which leads to more accurate comparisons when outcomes are aggregated over repeated samples. However, in the online learning to rank setting with live data only, ranker comparisons are based on a single data sample, and information about the magnitude of ranker differences is lost. Developing online learning methods that exploit this additional information is a direction for future work.

Under the *almost random* click model, where the level of noise is highest, the online performance achieved under BI, TD, and PI is equivalent. In one case (*TD2003*, row 32), PI even improves significantly over the baseline TD. Examining the offline performance for PI under this noise model (shown for *NP2003* in Figure 7.1(d)), we see that PI performs significantly higher than BI and TD. This suggests that learning with PI is more robust to noise than when using the other baseline methods. Most likely, the lack of fidelity shown for these methods in Chapter 4 gets magnified under noise, leading to slower learning for these methods.

Our methods that learn from historical data are enabled by PI, meaning that to improve performance over the best-performing baseline, TD, the methods need to learn substantially faster, to overcome the initial performance loss incurred by the randomization inherent to PI. For the CPS method, we see that this is the case. The method achieves much higher online performance than any of the methods that learn with live data only. Furthermore, the performance gain for reusing historical data is particularly big when click feedback is noisy, with gains of up to 104.5% under the *informational* click model (*HP2004*, row 20), and up to 153.4% under the *almost random* click model (*NP2003*, row 30). Looking at offline performance (Figure 7.1), we observe that CPS learns faster than methods that learn from live data only. In particular, the speed-up increases with noise in the click model, suggesting that CPS can effectively limit the effect of noise in click feedback.

Performance for RHC is generally lower than that obtained by CPS or the baselines that only take live data into account. However, with bias correction (RHC-U) and under noisier feedback, performance of this method increases. First, under all click models, we see that RHC-U generally outperforms RHC-B. This performance improvement is statistically significant in most cases (rows 1–8, 10–15, 19–23, 26, and 28–32). In only two cases, RHC-U performs significantly worse than RHC-B (on *OHSUMED*, rows 25 and 34), and in the remaining cases, the performance of the two variants of RHC is equivalent. This performance improvement under RHC-U indicates the importance sampling component used to correct for bias is effective. The resulting unbiased estimates of comparison outcomes under historical data not only provide a good estimate of the relative ranker quality, but also indicate how reliable these estimates are so that they can be properly combined with live estimates.

Looking at the performance of RHC under the *perfect* click model, we see that its performance is significantly worse than for the baseline TD on all data sets. Under the slightly noisier *navigational* click model, the performance of RHC-B is still worse than the baseline for most data sets, but that of RHC-U is equivalent to the baseline on six data sets. Under the *informational* click model, performance of RHC-U improves significantly over the baseline for five data sets (rows 19–24). Finally, under the *almost random* click model, the online performance of both variants of RHC improves significantly over the baseline for five data sets (rows 28–32), and is statistically equivalent for the remaining four data sets (rows 33–36). Comparing again to offline performance, we



(a) *perfect* (b) *navigational* (c) *informational* (d) *almost random*

Figure 7.1: Offline performance of the baseline and RHB and CPS online learning to rank methods on the data set *NP2003* in terms of NDCG after up to 1,000 user impressions with varying click models.

find that RHC learns as quickly as the baseline methods, but cannot make as effective use of the learned rankers, similar to PI. However, when feedback is noisy, the method does succeed in making comparisons more reliable (cf., Figure 7.1(c)–7.1(d)). When feedback is at its noisiest, even the biased method RHC-B learns significantly faster than any of the baseline methods. Thus, we conclude that RHC can effectively leverage historical data to make ranker comparisons more reliable, but this results in performance gains only when click feedback is indeed noisy.

For CPS, the biased version of the method performs slightly better than CPS-U under *perfect* clicks (the differences are statistically significant in 4 cases, in rows 1, 3–4, and 6). However, as feedback becomes noisier, these differences become smaller and fewer differences are statistically significant. Under the *navigational* click model, results for three data sets are statistically significant (rows 11–13), under the *informational* click model, this is true for two data sets (rows 20 and 22), and under the *almost random* click model, this is true for only one data set (row 34). It appears that, in contrast to RHC, where historical estimates of interleaved comparison outcomes are combined with live estimates, accurately compensating for bias in CPS does not lead to further performance improvements. Instead, correcting for bias using importance sampling increases the noise of estimates, which may cause small drops in performance, especially when feedback is very reliable otherwise. While this trend can be observed in our results, performance differences between CPS-B and CPS-U are small. Therefore, the differences between these two variants of CPS should be explored in more detail in the future.

Our main results show consistently high performance for our CPS method, which can achieve significantly and substantially higher online performance than all other methods tested. We find that reusing historical data using CPS allows faster learning than with current online learning to rank methods that take only live data into account. For RHC, we found that the method can reduce noise and improve online performance substantially, but only when click feedback is noisy. In the next section, we analyze our results in more detail.

7.4 Analysis

In this section, we first compare the performance of our methods to supervised learning to rank approaches (§7.4.1). Then we compare our methods' sensitivity to parameter settings (§7.4.2 and 7.4.3).

7.4.1 Offline Performance

Most previous work on learning to rank for IR focused on supervised approaches, and measured the offline performance achieved by learners after all training data had been processed. Our approach is fundamentally different, as it learns online, from relative feedback observed on the result lists presented to users. Despite this more limited form of feedback, we showed in Chapter 6 that effective learning is possible. The algorithms developed in this chapter further improve on the learning speed of baseline learning algorithms.

To allow for some comparison with supervised learning to rank approaches, we show the offline performance achieved by CPS-U in terms of NDCG at different cutoffs on the *perfect* and *informational* click models in Table 7.2.[3] Note that this implementation of NDCG differs from that used in the LETOR benchmark (as discussed in §3.5), however at cutoff 1 (NDCG@1) the two metrics are equivalent.

Performance of CPS is competitive with the supervised learning to rank approaches included in the LETOR benchmark (Liu, 2009). For all included data sets, CPS with *perfect* feedback beats a simple regression approach. In addition, CPS beats more than half the included (supervised pairwise and listwise) approaches in terms of NDCG@1 on the data sets *HP2003*, *NP2004*, and *TD2003*. On one of these, *NP2004*, CPS improves over the offline performance of all supervised methods reported in (Liu, 2009) (best NDCG@1 achieved there is 0.533, while CPS achieves an NDCG@1 of 0.566). This demonstrates that competitive offline performance can be achieved by CPS, despite

[3]Results differ slightly from those reported in Hofmann et al. (2013a), as some errors were corrected. The corrected results are better, but show the same trend (in terms of the relative performance of the method under *perfect* and *informational* feedback).

	perfect			informational		
	N@1	**N@3**	**N@10**	**N@1**	**N@3**	**N@10**
HP2003	0.701	0.740	0.771	0.673	0.711	0.742
HP2004	0.590	0.682	0.732	0.529	0.613	0.663
NP2003	0.541	0.657	0.712	0.533	0.643	0.699
NP2004	0.566	0.682	0.738	0.535	0.647	0.700
TD2003	0.326	0.301	0.296	0.262	0.263	0.266
TD2004	0.402	0.359	0.309	0.291	0.272	0.239
OHSUMED	0.484	0.454	0.424	0.437	0.407	0.385
MQ2007	0.321	0.327	0.364	0.288	0.295	0.333
MQ2008	0.348	0.389	0.486	0.309	0.350	0.454

Table 7.2: Offline performance after 1,000 iterations in terms of NDCG and cutoffs 1, 3, and 10 for CPS-U under the *perfect* and *informational* click models.

the limited feedback, after only 1,000 iterations. Further improvements are possible for longer run times.

The offline performance of CPS remains relatively high under the *informational* click model. The reason is that the method compensates for some of the click noise. The biggest drop in NDCG@1 is observed for the data set *TD2004*, with a decrease in offline performance by 28% (from 0.427 to 0.307). The smallest decrease is observed for the data set $NP2003$. There, offline performance under the *informational* click model is only 1.5% lower (0.533) than under *perfect* feedback (0.541). Overall, our results show that good offline performance can be achieved by CPS, even when feedback is noisy.

7.4.2 CPS – Sensitivity to Parameter Settings

Above, we reported results for only one set of parameters. Here, we investigate the sensitivity of CPS to changes in these parameters. CPS has three parameters: the history length λ (default: 10), the size of the candidate pool η (default: 6), and the number of historical comparisons per candidate pair ζ (default: 10). The algorithm is linear in η and ζ per live update ($O(\eta\zeta)$). An increase in λ does not significantly affect the run time of the algorithm, but determines the number of historical samples kept in memory, from which the samples for candidate comparisons are selected.

Figure 7.2 (parts a–c) shows the online performance achieved by CPS-U under the navigational click model on the data set *NP2003* when varying one parameter at a time. The online performance of CPS-U in this setting with default parameter settings is 102.97, as shown in Table 7.1 (row 12). Decreasing η, the size of the candidate pool, to $\eta = 2$ leads to a decrease in performance of 12.3% percent (to 90.29). Increasing the number of candidates to 10 increases online performance to 104.24, a much smaller change of 1.2%. This suggests that the performance reported above (§7.3) can be further increased by using larger candidate pools. However, returns are expected to diminish as ever more candidates are used.

For the number of repetitions performed to compare candidate rankers using historical data (ζ, default setting: $\zeta = 10$), effects are much smaller. We observe a small change

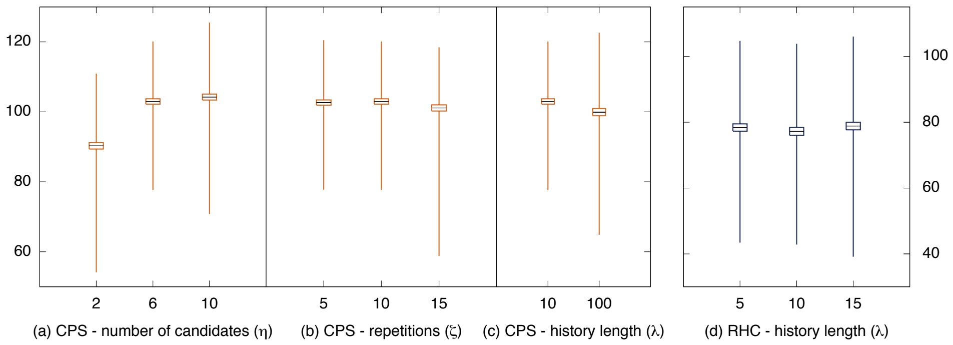


Figure 7.2: Online performance (min, max, and mean with standard error) after 1,000 iterations for different settings of parameters (a) η, (b) ζ, and (c) λ for CPS and (d) λ for RHC on the *NP2003* data set and the *navigational* click model. (Note the differences in scale of the y-axes between plots (a)–(c) and (d).)

in mean online performance when changing the setting to $\zeta = 5$, but the change is not statistically significant. Increasing the number of repetitions to $\zeta = 15$ results in a small but significant drop in performance, likely because additional comparisons increase noise without providing additional information. These results suggest that investing additional computational resources in increasing ζ is less beneficial than increasing η, as shown above.

Increasing the history length to $\lambda = 100$ significantly decreases the performance of CPS-U. The reason is that the more recent historical samples used with a smaller λ are collected on ranker pairs that are more similar to the current candidate rankers. When older samples are used instead, the variance of historical outcome estimates increases (under CPS-B, bias would increase), leading to diminished performance.

Overall, we find that performance under CPS can be further improved by increasing the size of the candidate pool. For the remaining parameters, performance is relatively stable and decreases gracefully when less optimal settings are used. Finally, our analysis indicates that additional computational resources are best spent on increasing the size of the candidate pool (η). Although the linear increase in computation is expected to lead to sub-linear performance gains, developers of deployed applications are typically willing to invest in additional computing time when it translates to even small performance gains (while in a scientific setting computational resources limit what experiments are feasible to run).

7.4.3 RHC – Sensitivity to Parameter Settings

RHC has only one parameter, the history length λ (cf., Algorithms 5 and 12, default: 10). This parameter determines how many historical data points are kept in memory, and are used to compare the current best ranker $\mathbf{w}_t$ to the candidate ranker $\mathbf{w}'_t$. This method is linear in λ per live update.

The sensitivity of RHC-U to changes in λ is shown in Figure 7.2, part (d). Setting

$\lambda = \{5, 15\}$ has no significant effect on online performance. The slightly lower performance of the original setting of $\lambda = 10$ is likely due to noise. We can conclude that the performance of this algorithm is relatively robust to changes in λ (thus, investing in additional resources to increase λ is not recommended).

7.5 Conclusion

In this chapter, we investigated whether and how historical data can be reused to speed up online learning to rank for IR. We proposed two approaches for integrating estimates based on historical data with a stochastic gradient descent algorithm for online learning to rank. Our first approach, RHC, uses historical comparison estimates to complement live comparisons and to make them more reliable. Our second approach, CPS, uses historical data for preselecting candidate rankers, thereby improving the quality of the rankers that are evaluated in live interactions with search engine users.

Our experimental evaluation of the proposed methods, based on the nine LETOR data sets and four click models that allowed us to investigate online performance of the methods under varying levels of click noise, yielded several insights. First, we found that CPS can substantially and significantly speed up online learning to rank for IR. We observed high gains in online performance over methods that use live data only for all click models. Second, performance gains of CPS were particularly high when click feedback was noisy. This result demonstrates that CPS is effective in compensating for noise in click feedback. Third, RHC was found to make ranker comparisons more reliable. However, positive effects on learning were observed only under noisy feedback and performance gains were lower than those obtained by CPS. Finally, we found that compensating for bias in click feedback substantially improved the performance of RHC, where historical estimates of interleaving outcomes are combined with live outcomes, but had small (negative) effects on the performance of CPS.

This work is the first to show that historical data can be used to significantly and substantially improve online performance in online learning to rank for IR. It also demonstrates that our interleaved comparison methods PI-MA and PI-MA-IS open up new directions for collecting and using interaction data in online learning to rank for IR. Interestingly, best results were obtained by improving the quality of the candidate rankers using CPS. This finding suggests that developing more complex sampling and exploration schemes is a promising direction for follow-up work.

This chapter concludes our investigation of the principles under which online learning to rank for IR can be reliable and efficient. In the next chapter (Chapter 8), we draw conclusions from all research chapters and present our main findings and directions for future work.

8 Conclusions

In this thesis, we presented work towards enabling "self-learning search engines" that can automatically adjust to user behavior and preferences. Broadly, we investigated whether search engines can learn effectively from user interactions. Online learning to rank for IR is different from the most commonly used supervised learning approaches, because user interactions with such a system are most suitable for interpretation as relative feedback, and because feedback can be noisy and biased (e.g., due to the order in which search results are displayed). In addition, online learning to rank algorithms need to learn quickly from the available user interactions, and they need to make use of what has been learned as well as possible to satisfy users' expectations even while learning.

The four research chapters of this thesis addressed the challenges of online learning to rank for IR as follows. First, in Chapter 4, we focused on how to extract information that is as useful as possible from the noisy, biased user interactions that such a system can observe. In particular, we analyzed interleaved comparison methods, which allow comparisons between rankers using click feedback, and proposed new methods to address limitations of existing methods. Second, in Chapter 5, we investigated the limitations of click data and interleaved comparison methods in a real-live application, web search. Here, we focused on the effects search result presentation may have on user clicks, and how such effects can influence the outcomes of interleaving experiments. In Chapter 6, we turned to the online performance of online learning to rank for IR approaches, and tested the hypothesis that balancing exploration and exploitation can improve online performance in pairwise and listwise online learning to rank. Finally, in Chapter 7, we focused on the reliability and speed of learning in the online setting. Building on the interleaved comparison methods developed in Chapter 4, we developed two approaches for speeding up learning by reusing previously observed interaction data.

Below, we provide a more detailed summary of the contributions and results of our research, and answer the research questions set out at the beginning of this thesis (§8.1). We conclude with an outlook on future research directions (§8.2).

8.1 Main Findings

The starting point of this thesis was the earlier finding that implicit user feedback in a search setting is most reliable when interpreted relative to the presented rankings (Radlinski et al., 2008b). In particular, interleaved comparison methods had previously been shown to be able to compare result rankings using click data (Radlinski and Craswell, 2010). The first research questions we addressed in Chapter 4 focused on formalizing the characteristics of these methods:

RQ 1 What criteria should interleaved comparison methods satisfy?

RQ 2 Do current interleaved comparison methods satisfy these criteria?

To answer the first question, we proposed a framework for analyzing interleaved comparison methods in terms *fidelity*, *soundness*, and *efficiency*. Fidelity captures whether the expected comparison outcomes of an interleaved comparison method reflect differences in ranking quality as appropriate for an IR experiment. Soundness reflects whether the estimator of the method produces unbiased and consistent estimates of that expected outcome. Efficiency reflects the number of data samples required by the method. This framework allows more formal and systematic comparisons between interleaved comparison methods than was previously possible.

Using the proposed framework, we analyzed all existing interleaved comparison methods, and found that none exhibited fidelity. This means that for each method, there are cases where the expected outcome of the method does not reflect ranking quality appropriately.

To address these shortcomings, we designed probabilistic interleave (PI), which takes into account the magnitude of differences between rankings. We also designed an extension, PI-MA, to increase efficiency of the method.

RQ 3 Do PI and its extension PI-MA exhibit fidelity and soundness?

RQ 4 Is PI-MA more efficient than previous interleaved comparison methods? Is it more efficient than PI?

We showed analytically that PI and PI-MA exhibit fidelity and soundness. Then, we empirically compared the efficiency of PI-MA to existing interleaved comparison methods and to PI. We found PI-MA to be more efficient than existing methods, and more efficient than PI, which confirmed our analytical results.

We further extended PI to allow the estimation of interleaved comparison outcomes from historical data that was collected during earlier comparisons of other (source) rankers. This extension, PI-MA-IS, combines probabilistic interleaving with importance sampling. Reusing historical data for interleaved comparisons was previously not possible, resulting in the following questions:

RQ 5 Can historical data be reused to compare new ranker pairs?

RQ 6 Does PI-MA-IS maintain fidelity and soundness?

RQ 7 Can PI-MA-IS reuse historical data effectively?

The central question was whether it was possible to reuse data for interleaved comparisons. By showing that PI-MA-IS can indeed reuse such data, we were able to answer this question affirmatively. We analytically showed that PI-MA-IS is sound, as it maintains both fidelity and soundness under data that was collected following a distribution different from that would be obtained under the target ranker pair to be compared. We showed empirically that PI-MA-IS is the only interleaved comparison method that can effectively reuse historical data.

In Chapter 5 we turned to questions about how user clicks may be affected by result presentation, and how this may affect interleaving results:

RQ 8 (How) does result presentation affect user clicks (caption bias)?

RQ 9 Can we model caption bias, and compensate for it in interleaving experiments?

RQ 10 (How) does caption bias affect interleaving experiments?

We confirmed that result presentation has a strong effect on click behavior. We proposed and trained models of such caption bias on usage data of a web search engine, and found that click behavior was best explained by a combination of relevance, position, and caption features. The most important caption features included whether a result was presented with deep links, the amount of highlighting in the title, and the snippet length. Also, we found that per-document features had a much stronger affect on click behavior than document-pairwise features (such as whether a result title had more or less highlighting than the document ranked immediately before or after it). Building on our probabilistic caption-bias models, we developed extensions of two interleaved comparison methods, TD and PI-MA, that compensate for caption bias. In applying our model of caption bias to six real-life interleaving experiments, we found first evidence that caption bias can affect the outcome detected in an experiment, e.g., if the experiment includes rankers that use click data during training.

After developing and investigating new approaches and models for interleaved comparisons, we focused on principles of learning from such feedback inferred from user behavior. When learning from user feedback in an online setting, systems need to ensure that high-quality results are presented to satisfy user expectations, but also that the presented results ensure that high-quality feedback can be collected for future learning. In Chapter 6 we formulated this *exploration-exploitation dilemma* for online learning to rank, and addressed the following research questions:

RQ 11 Can balancing exploration and exploitation improve online performance in online learning to rank for IR?

RQ 12 How are exploration and exploitation affected by noise in user feedback?

RQ 13 How does the online performance of different types (pairwise and listwise) of online learning to rank for IR approaches relate to balancing exploration and exploitation?

We developed the first two approaches (one pairwise, one listwise) for balancing exploration and exploitation in an online learning to rank setting, and found that such a balance can substantially and significantly improve online performance. We found important differences between the two developed approaches and under different levels of noise in click behavior. While the original (purely exploitative) pairwise approach performs very well under perfect click feedback, it cannot learn from noisy click data. Adding exploration could partially compensate for this performance loss. The original (purely exploratory) listwise learning approach was found to over-explore under all levels of noise, and its performance could be significantly improved by balancing exploration and exploitation.

In Chapter 7 we combined our interleaved comparison methods PI-MA and PI-MA-IS to investigate whether reusing historical data for interleaved comparison methods could be used to improve online learning.

RQ 14 Can previously observed (historical) interaction data be used to speed up online learning to rank?

RQ 15 Is historical data more effective when used to make comparisons more reliable (as in RHC), or when used to increase local exploration (as in CPS)?

RQ 16 How does noise in user feedback affect the reuse of historical interaction data for online learning to rank?

We found that historical data could indeed be used to significantly and substantially improve online learning to rank for IR. While a straightforward application of this data to make comparisons more reliable (using RHC) resulted in moderate performance gains, using this data for preselecting new candidate rankers resulted in much higher gains. Performance gains were highest under noisy click data.

This thesis resulted in insights and algorithms for enabling large-scale online learning to rank for IR. The software developed for our experiments, along with reference implementations of the developed interleaved comparison and online learning to rank methods is available online (see Appendix A for details).

The goal of this thesis is to develop a better understanding of how search engines can learn from user interactions, and to translate this understanding into new algorithms that allow more effective learning in an online learning to rank setting. We advanced towards this goal in several ways. We now better understand how rankers can be compared using interleaved comparisons methods, and what the properties of these methods are. This understanding was translated into a new set of interleaved comparison methods that naturally take into account differences between result rankings, and that allow ranker comparisons using data that was collected with other rankers. Also, we better understand how the presentation of search results can affect user clicks, and we have developed a model for measuring and compensating for those clicks in interleaved comparisons. We learned to improve online performance of online learning to rank for IR by balancing exploration and exploitation, and found that such a balance is particularly helpful in listwise learning. Finally, we showed that it is possible to speed up online learning to rank for IR by reusing previous interactions for exploring candidate rankers.

The obtained results show that online learning to rank for IR can be efficient and effective. We expect them to have an impact both on the theoretical development of online learning to rank approaches, and on their practical applications. Our methods enable online learning to rank in practice, and we hope that they will contribute to practical search applications in the future. Many search settings have been addressed much less thoroughly than web search, and achieving good search performance in settings such as enterprise search or search of personal collections is notoriously difficult. These are the settings that may most immediately benefit from our results. Another early application is recency search, where first benefits of online learning have been demonstrated (Moon et al., 2012). More immediately, our results on interleaved comparison methods is directly applicable to online evaluation in a manual tuning setting. Using these methods, rankers tuned using e.g., expert knowledge or supervised learning, can be compared on real user interactions to assess their quality from a user perspective. However, in the long term, we expect that complete online learning to rank solutions will be an important solution to addressing situational relevance in a scalable manner.

Finally, mechanisms for learning from user interactions are much more broadly applicable than in an IR setting alone. Until now, these adaptations are often designed manually, with limited adaptation online. Figuring out how to develop systems that can automatically adapt to users by online learning is a major challenge. The principles identified here constitute a step towards that goal. For example, human feedback may be noisy, and may be more easily interpreted as relative than absolute in many settings, requiring effective algorithms for learning from such feedback. Similarly, many settings where computers learn directly from their users may require a focus on online performance, and a balance of exploration and exploitation. When these challenges are addressed, online learning to rank could facilitate smarter and more natural interactions between computers and human users.

8.2 Future Work

This thesis resulted in insights and algorithms for enabling online learning to rank for IR. Beyond these, it opens up many interesting and important directions for future work. Below, we outline three main areas: real-live applications, smarter exploration, and long-term learning and planning.

Applications. One important direction for following up on this work is to develop applications that put the developed methods to work in real search settings. As online learning to rank methods are only at the beginning of their development, many questions need to be addressed when making the transition to real-live applications. Within this thesis, we focused on principles of obtaining feedback and learning from it in an online learning to rank setting, independent of the particular search context in which these principles and the resulting technology would be applied. A first step in applying these principles is to identify sets of ranking features that can provide a good basis for online learning to rank for a variety of IR settings. Many features have been explored for supervised learning to rank. These may be similarly effective in the online learning setting. Additional relevant work has been done in the area of recency search, and features that have been found to effectively capture temporal aspects of query-document relationships may be applicable.

An orthogonal problem is to determine to what groups of users, queries, or higher-level tasks learning to rank should be applied. In our experiments, we abstracted from such fine-grained differences and instead applied online learning to rank with the goal of identifying global patterns of ranking quality. However, when a system has many users, large amounts of interaction data can be observed and used to learn rankings for relatively small groups. Positive effects of personalization have been demonstrated in, e.g., web search (Matthijs and Radlinski, 2011; Teevan et al., 2008), and we therefore think that adapting to the preferences of individuals, or small groups of users, can further improve the performance of search engines that use online learning to rank. Similarly, online learning has been demonstrated to provide positive effects when adapting to individual frequent queries, as well as to global ranking preferences (albeit in a setting with absolute feedback (Moon et al., 2012)). Challenges that need to be addressed include finding effective similarity metrics for identifying appropriate levels of adaptation and designing mechanisms for interpolating between ranking functions learned for different levels of granularity.

Our methods have been evaluated using simulated user interactions, except in Chapter 5, where we studied effects of search result presentation on user clicks. For studying principles of online learning and developing new algorithms, our setup had the advantage that it enabled experiments in various settings, such as different amounts of noise in click feedback. Naturally, assumptions underlying these simulations need to be tested before moving these methods to a real setting. In addition to validating our results in more realistic settings, such additional experiments can help develop more complex models of, e.g., user interaction that can be applied to the online learning to rank setting. Interesting areas include the application of the most recent click models or exploring the use of implicit feedback beyond clicks.

Smart exploration. A second area of development is to further improve our understanding of the fitness landscape in an online learning to rank for IR setting. In some aspects, this area is similar to the supervised learning setting, where optimizing for the typically non-smooth and non-differentiable IR metrics proved to be hard, and several approaches have been developed to address this problem (such as approximations). The online learning to rank setting differs in that for many of the developed approaches it is not clear in how far they can be transferred to a setting where only relative feedback is provided for learning.

In this thesis, we demonstrated the importance of balancing exploration and exploitation in online learning to rank. A better understanding of the fitness landscape in this setting could help address the question of how best to explore. Current approaches for online learning to rank for IR are based on stochastic exploration. An advantage is that the resulting learning approaches make few assumptions about the structure of the feature and ranking solutions, and that they are very computationally efficient. However, their random exploration may result in many wasteful exploration steps. Algorithms that explore more systematically could substantially increase the speed of learning, and therefore online performance. Support is given by our results on improved candidate selection with CPS. By exploring candidate rankers more thoroughly, learning could be sped up and online performance improved significantly, particularly under noisy click feedback. Future exploration methods could model the effects of changes on individual features, or could try to explicitly model the fitness landscape. Obvious starting points for developing smarter exploration methods for listwise learning are methods for exploration in policy search RL (Heidrich-Meisner and Igel, 2009; Kalyanakrishnan et al., 2012). For the pairwise approach, the cost of random exploration is high. Exploration methods based on active learning approaches (Donmez and Carbonell, 2009; Tian and Lease, 2011; Xu et al., 2007, 2010) are a promising alternative that may allow more targeted exploration of promising areas of the solution space. This could reduce the cost of exploration, while maintaining or even improving long-term learning.

A limitation of our work is that a number of parameters, such as the exploration rates k and ϵ, and the learning step sizes α and δ in Chapter 6, of our learning methods were fixed, typically to values that had resulted in good performance in previous work. In supervised learning settings, such parameters would be tuned using cross-validation, but this is not a realistic option in an online learning to rank scenario where no labeled training data is provided. Results of our experiments with the nine LETOR 3.0 and 4.0 data sets showed that learning is effective even without additional tuning and that our findings are stable across data sets and tasks. However, it is likely that better performance can be obtained with additional tuning, both to a particular online learning to rank setting and to changes in user behavior and preferences over time (in non-stationary settings). Consequently, an interesting question is whether and how such a system could automatically adjust its learning parameters to a specific setting.

Long-term planning and learning. A third direction for following up on the work presented in this thesis is to explore and develop algorithms that adapt to long-term interactions with their users. So far, we have modeled the interactions between search engine and user as a contextual bandit problem. Relaxing the assumption of independence between system actions and queries makes the problem much more difficult to address, but poses opportunities for a diverse range of interaction patterns.

One direction of development is to model interactions between user and search engine as a full Markov Decision Process (MDP) (Sutton and Barto, 1998), where the agent's actions (e.g., retrieval results) can affect the state of the environment (e.g., the cognitive state of the user). This could enable systems that learn to guide users paths through information spaces, by building up towards more complex material, or by adding interactions with the searcher that can lead to new associations between concepts. Formulating interactions as a partially observable MDP (POMDP) (Kaelbling et al., 1998), would allow the system to infer states (of the user) when information is partially hidden, similar to a recent application in the related domain of ad-selection (Yuan and Wang, 2012). While this and other existing solutions do not address the relative feedback setting, extensions towards long-term online learning to rank for IR could make use of preference-based RL methods that are being developed in the RL community (Fürnkranz et al., 2012). The resulting methods can pave the way towards more effective long-term interactions within search sessions, over tasks, or over even longer periods of time.

APPENDIX

A Software

The software used to run the experiments described in Chapters 4, 6, and 7 is made available as the online learning framework OL2R on `http://ilps.science.uva.nl/resources/online-learning-framework`. OL2R contains all software required to run the experiments, following the experimental setup detailed in Chapter 3.

Below, we describe the prerequisites for running OL2R (§A.1), and give short step-by-step instructions for running evaluation and learning to rank experiments (§A.2). Then, we detail the contents of the package, and finish with a brief outline of possible extensions of this software (§A.3).

A.1 Prerequisites

OL2R is implemented in python, and has the following prerequisites:

- **Python** - version 2.7 or higher
- **PyYaml**
- **Numpy** - version 1.6.1 or higher

A.2 Getting Started

OL2R is provided as an archive in tar format on `http://ilps.science.uva.nl/resources/online-learning-framework`. After downloading and extracting the archive, two types of experiments can be run: online evaluation experiments replicate the experimental setup of Chapter 4, and online learning experiments replicate the experimental setup of Chapters 6–7.

Setting up and running both evaluation and learning experiments requires the following four steps:

1. prepare the data
2. create a configuration file
3. run the experiment
4. summarize experiment outcomes

We provide an example for evaluation experiments in §A.2.1, and an example for learning experiments in §A.2.2.

A.2.1 Running Evaluation Experiments

1. Data. OL2R accepts input data in SVMLight format.[1] The MSLR-WEB10K and LETOR 3.0 and 4.0 data sets are provided in the required format and can be obtained from `http://research.microsoft.com/en-us/projects/mslr/` and `http://research.microsoft.com/en-us/um/beijing/projects/letor/`, respectively. For example, download the MQ2008 data set of LETOR 4, and note the location of the data as `$DATA_DIR`. All tools accept input files in either plain text format, or compressed using `gzip`.

For evaluation experiments, data files need to be split in individual files per query – to enable fast randomization of the queries during the experiment. For this purpose, a script is provided with the package. It is called as follows:

Listing A.1: Splitting SVMLight data by query.

```
python src/python/split-query-file.py \
    $DATA_DIR/INPUT_FILE $DATA_DIR/MQ2008-SPLIT $FEATURE_COUNT
```

2. Configuration. To set up an evaluation experiment, prepare a configuration file in yml format.[2] For example, start from the template provided in Listing A.2, and save it as `config-eval.yml` (replace `DATA_DIR` and `OUTPUT_DIR` as appropriate).

Listing A.2: Example configuration for evaluation experiments.

```
query_dir: $DATA_DIR/MQ2008-SPLIT
feature_count: 46
num_runs: 10
num_queries: 1000
result_length: 10
# cascade model with 5 relevance grades
user_model: environment.CascadeUserModel
# for p-click and p-stop provide mappings from relevance grades to
# probabilities (here: perfect click model)
user_model_args:
  --p_click 0:0.0, 1:0.2, 2: 0.4, 3: 0.8, 4:1.0
  --p_stop 0:.0, 1:.0, 2:.0, 3:.0, 4:.0
# method names can be arbitrary strings and have to be unique
live_evaluation_methods:
- BI
- TD
- DC
- PI-MA
# provide arguments per method in matching order
live_evaluation_methods_args:
-  # BI
  --class_name comparison.BalancedInterleave
  --ranker ranker.DeterministicRankingFunction --ranker_args None
  --startinglist random
```

[1]For details, see `http://svmlight.joachims.org/`. A tool for converting whitespace or comma separated files to SVMLight format is available at `http://www.soarcorp.com/svm_light_data_helper.jsp`.

[2]PyYaml accepts input in Yaml version 1.1. See `http://yaml.org/spec/1.1/` for details.

```
-   # TD
  --class_name comparison.TeamDraft
  --ranker ranker.DeterministicRankingFunction --ranker_args None
-   # DC
  --class_name comparison.DocumentConstraints
  --ranker ranker.DeterministicRankingFunction --ranker_args None
  --startinglist random
-   # PI-MA
  --class_name comparison.ProbabilisticInterleave
  --ranker ranker.ProbabilisticRankingFunction --ranker_args 3
output_dir: $OUTPUT_DIR/experiment-1
output_prefix: evaluation-test-MQ2008
# set to False to avoid accidentally overwriting previous experiments
output_dir_overwrite: True
```

Note that all interleaved comparison methods that are compared to each other should be configured to run in the same experiment. This ensures that all methods are run on the same random sample of queries, which reduces variance in experiment outcomes.

3. Running the experiment. Evaluation experiments are run using configuration files as follows:

Listing A.3: Running evaluation experiments.

```
python src/python/evaluation-experiment.py -f config-eval.yml
```

The experiment output is stored in yml files in the output directory provided in the configuration file.

4. Summarizing experiment outcomes. The output files produced by an evaluation experiment can be summarized using the script provided with OL2R:

Listing A.4: Summarizing online evaluation experiments.

```
python src/python/summarize-evaluation-experiment.py --fold_dirs \
    $OUTPUT_DIR --metrics live_outcomes.BI live_outcomes.TD \
    live_outcomes.DC live_outcomes.PI-MA -t 1 5 10 50 100 500 1000 \
    --print_every 10 --output_base $OUTPUT_DIR/experiment-1
```

Results are aggregated over runs and folds, and arbitrarily many folds can be listed per experiment. Output is produced in the form of space separated files, one every `print_every` runs, that can be further processed using e.g., gnuplot. The produced files provide the accuracies and lower and upper bounds of the 95% binomial confidence intervals for all compared interleaved comparison methods after $\mathtt{t}_1 \ldots \mathtt{t}_n$ queries:

Listing A.5: File format of evaluation experiment summaries.

```
<line> .=. <query_count> <metric_mean> <lower_bound> <upper_bound>
     ... <metric_mean> <lower_bound> <upper_bound>
<query_count> .=. <integer>
<metric_mean> .=. <float>
<metric_lower_bound> .=. <float>
<metric_upper_bound> .=. <float>
```

A.2.2 Running Learning Experiments

1. Data. Data for online learning to rank experiments needs to be pre-processed to normalize feature values per query. When using LETOR data sets, a normalized version of the data is already provided in the *QueryLevelNorm* version of the data. For data sets that are not normalized per query, we provide a script for pre-processing. Like all other scripts provided with OL2R, this script accepts files in SVMLight format (as plain text or compressed using `gzip`).

Listing A.6: Normalizing data sets per query.

```
python src/python/normalize-per-query.py \
    $DATA_DIR/INPUT_FILE $DATA_DIR/OUTPUT_FILE $FEATURE_COUNT
```

2. Configuration. The configuration files for learning experiments are similar to those for evaluation experiments. For example, a test experiment on the MQ2008 data set can be configured as follows:

Listing A.7: Example configuration for learning experiments.

```
test_queries: $DATA_DIR/MQ2008/Fold1/test.txt
training_queries: $DATA_DIR/MQ2008/Fold1/Fold1/train.txt
feature_count: 46 # 64 for .Gov, 46 for MQ*, 136 for MSLR
num_runs: 10
num_queries: 500
# cascade model with 3 relevance grades
user_model: environment.CascadeUserModel
# for p-click and p-stop provide mappings from relevance grades to
# probabilities (here: perfect click model)
user_model_args:
    --p_click 0:.0, 1:1.0, 2:1.0
    --p_stop 0:.0, 1:.0, 2:.0
# baseline listwise learning system with team draft interleaving and
# deterministic rankers
system: retrieval_system.ListwiseLearningSystem
system_args: --init_weights zero --comparison comparison.TeamDraft
    --delta 1.0 --alpha 0.01 --ranker_tie random
    --ranker ranker.DeterministicRankingFunction
output_dir: $OUTPUT_DIR
output_prefix: MQ2008-Fold1
# set to False to avoid accidentally overwriting previous experiments
output_dir_overwrite: True
```

In contrast to online evaluation experiments, online learning to rank experiments are configured for one method at a time (here: baseline listwise learning system with TD).

3. Running the experiment. Learning experiments are run using configuration files as follows:

Listing A.8: Running online learning to rank experiments.

```
python src/python/learning-experiment.py -f config-learn.yml
```

4. Summarizing experiment outcomes. The output files produced by a learning experiment can be summarized using the script provided with OL2R:

Listing A.9: Summarizing online learning to rank experiments.

```
python src/python/summarize-learning-experiment.py --fold_dirs \
    $OUTPUT_DIR > $SUMMARY_FILE
```

Arbitrarily many folds can be listed per experiment. Results are aggregated over runs and folds. The output format is a space separated text file that can be further processed using e.g., gnuplot. The output files contain the mean and standard deviation of the offline and online performance after `n` queries.

Listing A.10: File format of learning experiment summaries.

```
<line> .=. <query_count> <offline_mean> <offline_stddev> \
           <online_mean> <online_stddev>
<query_count> .=. <integer>
<offline_mean> .=. <float>
<offline_stddev> .=. <float>
<online_mean> .=. <float>
<online_stddev> .=. <float>
```

A.3 Package Contents and Extensions

Apart from the scripts demonstrated above, the OL2R consists of 8 packages that implement its functionality as follows:

comparison	interleaved comparison methods for comparing rankers using click data; contains the baseline interleaved comparison methods described in §2.3.1, the probabilistic interleave methods developed in Chapter 4, and the RHC method developed in Chapter 7
environment	click models for simulating user interactions
evaluation	evaluation metrics (NDCG)
experiment	entry level classes for learning and evaluation experiments
query	parse and provide access to collections of queries (with document features and relevance judgments)
ranker	deterministic and probabilistic ranking functions
retrieval_system	online learning retrieval systems, e.g., for pairwise and listwise learning to rank; contains the baseline learner DBGD (§2.5.2), the methods for balancing exploration and exploitation developed in Chapter 6, and the CPS method developed in Chapter 7
utils	various utility functions

The code is intended to be extended with new interleaved comparison methods and methods for online learning to rank for IR. The most obvious points for extension are:

- comparison – extend AbstractInterleavedComparison to add new interleaving or inference methods.
- environment – extend AbstractUserModel to enable evaluation under different assumptions about user behavior.
- evaluation – extend AbstractEval to add evaluation metrics.
- retrieval_system – extend AbstractLearningSystem to add a new mechanism for learning from click feedback.

Bibliography

D. Agarwal, B. Chen, P. Elango, N. Motgi, S. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS '08*, pages 17–24, 2008. (Cited on page 25.)

E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pages 19–26, 2006. (Cited on pages 17 and 18.)

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003. (Cited on page 24.)

P. Bailey, N. Craswell, A. de Vries, and I. Soboroff. Overview of the TREC 2007 enterprise track. In *TREC 2007*, volume 7, 2007. (Cited on page 11.)

K. Balog, P. Serdyukov, and A. de Vries. Overview of the TREC 2011 entity track. In *TREC 2011*, 2011. (Cited on page 11.)

A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, 40:201–211, 1981. (Cited on pages 24, 25, and 32.)

N. Belkin, R. Oddy, and H. Brooks. ASK for information retrieval: Part i. background and theory. *Journal of Documentation*, 38(2):61–71, 1982. (Cited on page 10.)

P. Bennett, F. Radlinski, R. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *SIGIR '11*, pages 135–44, 2011. (Cited on page 11.)

K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *ECIR '10*, pages 13–25, 2010. (Cited on page 11.)

J. Besser, M. Larson, and K. Hofmann. Podcast search: User goals and retrieval technologies. *Online Information Review*, 34(3):395–419, 2010. (Cited on pages 1, 7, and 11.)

J. Boyan, D. Freitag, and T. Joachims. A machine learning architecture for optimizing web search engines. In *AAAI '96 Workshop on Internet Based Information Systems*, pages 1–8, 1996. (Cited on page 17.)

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107 – 117, 1998. (Cited on page 11.)

A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002. (Cited on page 34.)

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05*, pages 89–96, 2005. (Cited on pages 12 and 15.)

C. Burges, R. Ragno, and Q. Le. Learning to rank with non-smooth cost functions. In *NIPS '06*, 2006. (Cited on page 15.)

C. Burges, K. Svore, P. Bennet, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *Journal of Machine Learning Research: Workshop and Conference Proceedings*, 14: 25–35, 2011. (Cited on page 15.)

V. Bush. As we may think. *The Atlantic Monthly*, 1945. (Cited on page 9.)

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07*, pages 129–136, 2007. (Cited on pages 13 and 15.)

D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *CIKM '09*, pages 1227–1236, 2009. (Cited on page 11.)

B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS '07*, pages 217–224. MIT Press, 2008. (Cited on pages 17 and 18.)

O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09*, pages 1–10, 2009. (Cited on pages 17 and 18.)

O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*, pages 621–630. ACM, 2009. (Cited on pages 12, 18, and 34.)

O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems*, 30(1):6:1–6:41, 2012. (Cited on pages 18, 19, 20, 21, 42, 44, 45, 46, 48, 58, 61, 74, and 79.)

J. Chen, B. Xin, Z. Peng, L. Dou, and J. Zhang. Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(3):680–691, 2009. (Cited on page 25.)

Y. Chen. Another look at rejection sampling through importance sampling. *Statistics & probability letters*, 72 (4):277–283, 2005. (Cited on page 58.)

C. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 web track. Technical report, NIST Special Publication, 2009. (Cited on page 11.)

C. L. Clarke, N. Craswell, I. Soboroff, and A. Ashkan. A comparative analysis of cascade measures for novelty and diversity. In *WSDM '11*, WSDM '11, pages 75–84, 2011. (Cited on page 12.)

C. L. A. Clarke, E. Agichtein, S. Dumais, and R. W. White. The influence of caption features on clickthrough patterns in web search. In *SIGIR '07*, pages 135–142, 2007. (Cited on pages 22, 73, 75, 76, and 77.)

J. D. Cohen, S. M. McClure, and A. J. Yu. Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):933–942, 2007. (Cited on page 26.)

D. Cossock and T. Zhang. Subset ranking using regression. In *COLT '06*, pages 605–619, 2006. (Cited on page 14.)

K. Crammer, Y. Singer, et al. Pranking with ranking. In *NIPS '01*, volume 14, pages 641–647, 2001. (Cited on page 14.)

N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*, pages 87–94, 2008. (Cited on pages 22 and 33.)

E. Cutrell and Z. Guan. What are you looking for?: an eye-tracking study of information usage in web search. In *CHI '07*, pages 407–416, 2007. (Cited on page 23.)

P. Donmez and J. Carbonell. Active sampling for rank learning via optimizing the area under the ROC curve. In *ECIR '09*, pages 78–89, 2009. (Cited on pages 15 and 132.)

P. Donmez, K. M. Svore, and C. J. Burges. On the local optimality of lambdarank. In *SIGIR '09*, pages 460–467, 2009. (Cited on page 15.)

Z. Dou, R. Song, X. Yuan, and J. R. Wen. Are click-through data adequate for learning web search rankings? In *CIKM '08*, pages 73–82, 2008. (Cited on page 17.)

M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *ICML '11*, pages 1097–1104, 2011. (Cited on pages 26 and 27.)

S. Dumais and E. Cutrell. Optimizing search by showing results in context. In *CHI '01*, pages 277–284, 2001. (Cited on page 23.)

G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *WSDM '10*, pages 181–190, 2010. (Cited on page 18.)

G. Dupret, V. Murdoch, and B. Piwowarski. Web search engine evaluation using click-through data and a user model. In *Workshop on Query Log Analysis*, 2007. (Cited on pages 17 and 18.)

G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08*, pages 331–338, 2008. (Cited on pages 83 and 91.)

M. Efron. Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology*, 62(6):996–1008, 2011. (Cited on page 11.)

S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168, 2005. (Cited on page 17.)

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001. (Cited on page 75.)

N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3):183–204, July 1989. (Cited on page 14.)

J. Fürnkranz and E. Hüllermeier. *Preference learning: An introduction*. Springer, 2010. (Cited on page 14.)

J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89:123–156, 2012. (Cited on page 133.)

E. Garfield. "science citation index" – a new dimension in indexing. *Science*, 144(3619):649–654, 1964. (Cited on page 11.)

J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979. (Cited on page 26.)

L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in WWW search. In *SIGIR'04*, pages 478–479, 2004. (Cited on page 22.)

F. Graybill and R. Deal. Combining unbiased estimators. *Biometrics*, 15(4):543–550, 1959. (Cited on page 116.)

Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *CHI '07*, pages 417–420, 2007. (Cited on page 22.)

F. Guo, L. Li, and C. Faloutsos. Tailoring click models to user goals. In *WSCD '09*, pages 88–92, 2009a. (Cited on pages 22, 33, and 34.)

F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*, pages 124–131, 2009b. (Cited on pages 22 and 33.)

P. R. Halmos. The theory of unbiased estimation. *Ann. Math. Statist.*, 17(1):34–43, 1946. (Cited on page 44.)

J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *CIKM '09*, pages 2029–2032, 2009. (Cited on pages 19, 21, 22, 39, 45, 48, 50, 58, 61, and 62.)

V. Heidrich-Meisner and C. Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML '09*, pages 401–408. ACM, 2009. (Cited on page 132.)

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN '99*, volume 1, pages 97–102, 1999. (Cited on pages 14 and 95.)

D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. *Research and Advanced Technology for Digital Libraries*, pages 515–515, 1998. (Cited on page 10.)

B. Hjørland. The foundation of the concept of relevance. *Journal of the American Society for Information Science and Technology*, 61(2):217–237, 2010. (Cited on page 12.)

K. Hofmann, K. Balog, T. Bogers, and M. de Rijke. Integrating contextual factors into topic-centric retrieval models for finding similar experts. In *SIGIR '08 Workshop on Future Challenges in Expertise Retrieval (fCHER)*, 2008. (Cited on pages 7 and 11.)

K. Hofmann, M. de Rijke, B. Huurnink, and E. Meij. A semantic perspective on query log analysis. In *Working Notes for the CLEF 2009 Workshop*, 2009a. (Cited on page 7.)

K. Hofmann, E. Tsagkias, E. Meij, and M. de Rijke. The impact of document structure on keyphrase extraction. In *CIKM '09*, pages 1725–1728, 2009b. (Cited on pages 7 and 11.)

K. Hofmann, K. Balog, T. Bogers, and M. de Rijke. Contextual factors for finding similar experts. *Journal of the American Society for Information Science and Technology*, 61(5):994–1014, 2010a. (Cited on pages 1, 7, and 11.)

K. Hofmann, B. Huurnink, M. Bron, and M. de Rijke. Comparing click-through data to purchase decisions for retrieval evaluation. In *SIGIR '10*, 2010b. (Cited on pages 7 and 18.)

K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR '11*, pages 251–263, 2011a. (Cited on pages 7 and 93.)

K. Hofmann, S. Whiteson, and M. de Rijke. Contextual bandits for information retrieval. In *NIPS '11 Workshop on Bayesian Optimization, Experimental Design, and Bandits*, Granada, 2011b. (Cited on page 7.)

K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*, pages 249–258, 2011c. (Cited on pages 7, 21, and 53.)

K. Hofmann, F. Behr, and F. Radlinski. On caption bias in interleaving experiments. In *CIKM '12*, pages 115–124, 2012a. (Cited on page 7.)

K. Hofmann, S. Whiteson, and M. de Rijke. Estimating interleaved comparison outcomes from historical click data. In *CIKM '12*, pages 1779–1783, 2012b. (Cited on pages 7 and 21.)

K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM '13*, 2013a. (Cited on pages 7 and 123.)

K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval Journal*, 16(1):63–90, 2013b. (Cited on pages 7 and 93.)

K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (to appear)*, 2013c. (Cited on page 7.)

S. Ieong, N. Mishra, E. Sadikov, and L. Zhang. Domain bias in web search. In *WSDM '12*, pages 413–422. ACM, 2012. (Cited on page 23.)

P. Ingwersen and K. Järvelin. *The turn: Integration of information seeking and retrieval in context*, volume 18. Springer, 2005. (Cited on page 12.)

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002. (Cited on pages 12 and 51.)

S. Ji, K. Zhou, C. Liao, Z. Zheng, G.-R. Xue, O. Chapelle, G. Sun, and H. Zha. Global ranking by exploiting user clicks. In *SIGIR '09*, pages 35–42, 2009. (Cited on page 17.)

T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002. (Cited on pages 14, 16, 17, 18, 21, 28, 29, and 95.)

T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*, pages 79–96. Physica/Springer, 2003. (Cited on pages 19, 20, 39, 97, and 118.)

T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems*, 25(2):7+, April 2007. (Cited on pages 17 and 25.)

S. Jung, J. L. Herlocker, and J. Webster. Click data as implicit relevance feedback in web search. *Information Processing & Management*, 43(3):791 – 807, 2007. (Cited on page 17.)

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998. (Cited on page 133.)

L. P. Kaelbling. *Learning in Embedded Systems*. MIT Press, Cambridge, Massachussets, 1993. (Cited on page 26.)

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artifical Intelligence Research*, 4(1):237–285, 1996. (Cited on page 23.)

S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. Pac subset selection in stochastic multi-armed bandits. In J. Langford and J. Pineau, editors, *ICML '12*, pages 655–662, 2012. (Cited on page 132.)

J. Kamps, M. Koolen, and A. Trotman. Comparative analysis of clicks and judgments for IR evaluation. In *WSCD '09*, pages 80–87, 2009. (Cited on pages 17 and 27.)

E. Kanoulas, B. Carterette, P. Clough, and M. Sanderson. Session track overview. In *TREC 2010*, 2010. (Cited on page 11.)

D. Kelly. Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval*, 3(1–2):1–224, 2009. (Cited on page 11.)

D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2): 18–28, 2003. (Cited on page 17.)

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999. (Cited on page 11.)

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS '08*, pages 817–824, 2008. (Cited on pages 24, 25, and 32.)

J. Langford, A. Strehl, and J. Wortman. Exploration scavenging. In *ICML '08*, pages 528–535, 2008. (Cited on pages 18, 25, and 26.)

E. L. Lehmann. *Elements of Large-Sample Theory*. Springer, Berlin, Germany, 1999. (Cited on page 44.)

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW '10*, pages 661–670, 2010. (Cited on page 25.)

L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM '11*, pages 297–306, 2011. (Cited on pages 25 and 26.)

R. Lippmann. Pattern classification using neural networks. *Communications Magazine, IEEE*, 27(11):47–50, 2002. (Cited on page 51.)

T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–331, 2009. (Cited on pages 1, 13, 14, 15, 32, 102, and 123.)

T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR '07*, 2007. (Cited on page 35.)

Y. Liu, M. Zhang, L. Ru, and S. Ma. Automatic Query Type Identification Based on Click Through Information Information Retrieval Technology. In *AIRS '06*, pages 593–600, 2006. (Cited on page 35.)

P. Lubell-Doughtie and K. Hofmann. Learning to rank from relevance feedback for e-discovery. In *ECIR '12*, 2012. (Cited on page 7.)

D. J. C. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press, Boston, MA, USA, 1998. (Cited on pages 27 and 57.)

A. Mahajan and D. Teneketzis. Multi-armed bandit problems. *Foundations and Applications of Sensor Management*, pages 121–151, 2008. (Cited on page 26.)

N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *WSDM '11*, pages 25–34, 2011. (Cited on pages 11 and 131.)

D. Metzler. *A Feature-Centric View of Information Retrieval*, volume 27. Springer, 2011. (Cited on page 10.)

D. Metzler and W. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing & Management*, 40(5):735–750, 2004. (Cited on page 11.)

T. Minka and S. Robertson. Selection bias in the LETOR datasets. In *SIGIR '08 Workshop on Learning to Rank for Information Retrieval*, pages 48–51, 2008. (Cited on page 96.)

A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27(1):2:1–2:27, 2008. (Cited on page 18.)

C. Mooers. The next twenty years in information retrieval; some goals and predictions. *American Documentation*, 11(3):229–236, 1960. (Cited on page 9.)

T. Moon, W. Chu, L. Li, Z. Zheng, and Y. Chang. An online learning framework for refining recency search results with user click feedback. *ACM Transactions on Information Systems*, 30(4):20:1–20:28, 2012. (Cited on pages 130 and 132.)

D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artifical Intelligence Research*, 11:199–229, 1999. (Cited on page 24.)

E. Morrison. Longitudinal study of the effects of information seeking on newcomer socialization. *Journal of Applied Psychology; Journal of Applied Psychology*, 78(2):173, 1993. (Cited on page 12.)

R. Nallapati. Discriminative models for information retrieval. In *SIGIR '04*, pages 64–71, 2004. (Cited on page 14.)

I. Ounis, C. Macdonald, and I. Soboroff. Overview of the TREC 2008 blog track. Technical report, NIST Special Publication: TREC, 2008. (Cited on page 11.)

P. Over. The TREC interactive track: an annotated bibliography. *Information Processing & Management*, 37 (3):369–381, 2001. (Cited on page 11.)

U. Ozertem, R. Jones, and B. Dumoulin. Evaluating new search engine configurations with pre-existing judgments and clicks. In *WWW '11*, pages 397–406, 2011. (Cited on page 18.)

J. Ponte and W. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281. ACM, 1998. (Cited on page 10.)

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *ICML '06*, 2006. (Cited on page 26.)

D. Precup, R. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *ICML '00*, pages 759–766, 2000. (Cited on pages 26 and 27.)

F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*, pages 667–674, 2010. (Cited on pages 2, 12, 18, 19, 21, 36, 42, 58, 118, and 128.)

F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*, 2013. (Cited on page 19.)

F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *AAAI '06*, pages 1406–1412, 2006. (Cited on page 22.)

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML '08*, pages 784–791. ACM, 2008a. (Cited on pages 18 and 25.)

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, pages 43–52, 2008b. (Cited on pages 17, 18, 19, 20, 25, 27, 39, 44, 46, 47, 58, 79, 97, 118, and 127.)

F. Radlinski, P. N. Bennett, and E. Yilmaz. Detecting duplicate web documents using clickthrough data. In *WSDM '11*, pages 147–156, 2011. (Cited on page 46.)

H. Robbins. Some aspects of the sequential design of experiments. *Bull. Am. Math. Soc.*, 58:527–535, 1952. (Cited on page 26.)

S. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, 1977. (Cited on page 10.)

S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04*, pages 42–49, 2004. (Cited on page 11.)

J. J. Rocchio. Relevance feedback in information retrieval. 1971. (Cited on page 16.)

D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW '04*, pages 13–19, 2004. (Cited on pages 1 and 35.)

I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(02):95–145, 2003. (Cited on page 17.)

G. Salton. Associative document retrieval techniques using bibliographic information. *J. ACM*, 10(4):440–457, 1963. (Cited on page 11.)

G. Salton. Mathematics and information retrieval. *Journal of Documentation*, 35(1):1–29, 1979. (Cited on page 10.)

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, Nov. 1975. (Cited on page 10.)

G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26 (11):1022–1036, Nov. 1983. (Cited on page 10.)

M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010. (Cited on pages 2, 11, 12, and 15.)

M. Sanderson and W. Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(13): 1444–1451, 2012. (Cited on page 9.)

T. Saracevic. Relevance: A review of the literature and a framework for thinking on the notion in information science. part ii: nature and manifestations of relevance. *Journal of the American Society for Information Science and Technology*, 58(13):1915–1933, 2007. (Cited on page 12.)

C. O. Schmidt and T. Kohlmann. When to use the odds ratio or the relative risk? *International Journal of Public Health*, 53:165–167, 2008. (Cited on page 79.)

F. Scholer, M. Shokouhi, B. Billerbeck, and A. Turpin. Using clicks as implicit judgments: expectations versus observations. In *ECIR '08*, pages 28–39, 2008. (Cited on page 17.)

D. Sculley. Large scale learning to rank. In *NIPS '09 Workshop on Advances in Ranking*, 2009. (Cited on pages 14, 28, 29, and 95.)

X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05*, pages 43–50, 2005. (Cited on pages 1, 11, 17, and 18.)

P. Shivaswamy and T. Joachims. Online structured prediction via coactive learning. In *ICML '12*, pages 1431–1438, 2012. (Cited on page 16.)

C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999. (Cited on pages 93 and 96.)

I. Soboroff and D. Harman. Overview of the TREC 2003 novelty track. In *TREC 2003*, pages 38–53, 2003. (Cited on page 11.)

K. Spärck Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management*, 36(6):779 – 808, 2000. (Cited on page 10.)

A. Strehl, C. Mesterharm, M. Littman, and H. Hirsh. Experience-efficient learning in associative bandit problems. In *ICML '06*, pages 889–896, 2006. (Cited on page 25.)

A. M. Strehl, J. Langford, L. Li, and S. M. Kakade. Learning from logged implicit exploration data. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *NIPS '10*, pages 2217–2225. 2010. (Cited on page 26.)

M. Strens. A Bayesian framework for reinforcement learning. In *ICML '00*, pages 943–950, 2000. (Cited on page 26.)

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS '00*, pages 1057–1063, 2000. (Cited on page 24.)

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988. (Cited on page 24.)

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA, USA, 1998. (Cited on pages 23, 26, 31, 32, 36, 51, 96, and 133.)

M. Szummer and E. Yilmaz. Semi-supervised learning to rank with preference regularization. In *CIKM '11*, pages 269–278. ACM, 2011. (Cited on page 15.)

M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM '08*, pages 77–86. ACM, 2008. (Cited on page 15.)

J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR '08*, pages 163–170, 2008. (Cited on pages 11 and 131.)

A. Tian and M. Lease. Active learning to maximize accuracy vs. effort in interactive information retrieval. In *SIGIR '11*, pages 145–154, 2011. (Cited on page 132.)

A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR '98*, pages 2–10, 1998. (Cited on page 23.)

E. Tsivtsivadze, K. Hofmann, and T. Heskes. Large scale co-regularized ranking. In *ECAI Workshop on Preference Learning: Problems and Applications in AI*, 2012. (Cited on pages 7 and 15.)

C. Van Rijsbergen. *The geometry of information retrieval.* Cambridge University Press, 2004. (Cited on page 10.)

E. Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of cross-language information retrieval systems*, pages 143–170. Springer, 2002. (Cited on page 11.)

E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval.* Digital Libraries and Electronic Publishing. MIT Press, 2005. (Cited on page 11.)

K. Wang, T. Walker, and Z. Zheng. PSkip: estimating relevance ranking quality from web search clickthrough data. In *KDD '09*, pages 1355–1364, 2009. (Cited on pages 17 and 23.)

C. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, 1989. (Cited on page 26.)

S. Whiteson and P. Stone. On-line evolutionary computation for reinforcement learning in stochastic domains. In *GECCO '06*, pages 1577–1584, July 2006a. (Cited on page 96.)

S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, 2006b. (Cited on page 26.)

T. D. Wilson, N. J. Ford, D. Ellis, A. E. Foster, and A. Spink. Information seeking and mediated searching: Part 2. uncertainty and its correlates. *Journal of the American Society for Information Science and Technology*, 53(9):704–715, 2002. (Cited on page 12.)

D. Xu, Y. Liu, M. Zhang, S. Ma, and L. Ru. Incorporating revisiting behaviors into click models. In *WSDM '12*, pages 303–312, 2012. (Cited on page 33.)

Z. Xu and R. Akella. A Bayesian logistic regression model for active relevance feedback. In *SIGIR '08*, pages 227–234, 2008. (Cited on page 15.)

Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *ECIR '07*, pages 246–257, 2007. (Cited on pages 15 and 132.)

Z. Xu, K. Kersting, and T. Joachims. Fast active exploration for link-based preference learning using gaussian processes. In *ECML PKDD'10*, pages 499–514, 2010. (Cited on pages 32 and 132.)

G. Xue, H. Zeng, Z. Chen, Y. Yu, W. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM '04*, volume 8, pages 118–126, 2004. (Cited on page 46.)

E. Yilmaz, M. Shokouhi, N. Craswell, and S. Robertson. Expected browsing utility for web search evaluation. In *CIKM '10*, pages 1561–1564, 2010. (Cited on page 12.)

S. Yuan and J. Wang. Sequential selection of correlated ads by POMDPs. In *CIKM '12*, pages 515–524, 2012. (Cited on page 133.)

Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML '09*, pages 1201–1208, 2009. (Cited on pages 12, 16, 19, 29, 30, 94, 97, 99, and 118.)

Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *SIGIR '10*, pages 507–514, 2010a. (Cited on pages 42 and 48.)

Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data. In *WWW '10*, pages 1011–1018, 2010b. (Cited on pages 22, 73, 74, 75, 78, 80, and 81.)

Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538 – 1556, 2012. (Cited on pages 16 and 19.)

J. Zhang and J. Kamps. A search log-based approach to evaluation. In *ECDL '10*, pages 248–260, 2010. (Cited on page 18.)

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML '04*, pages 116+. ACM, 2004. (Cited on pages 28 and 29.)

Y. Zhang, W. Xu, and J. Callan. Exploration and exploitation in adaptive filtering based on bayesian active learning. In *ICML '03*, pages 896–904, 2003. (Cited on page 25.)

F. Zhong, D. Wang, G. Wang, W. Chen, Y. Zhang, Z. Chen, and H. Wang. Incorporating post-click behaviors into a click model. In *SIGIR '10*, pages 355–362, 2010. (Cited on page 91.)

Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *WSDM '10*, pages 321–330, 2010. (Cited on page 91.)

Samenvatting

De hoeveelheid digitale data die we elke dag produceren is vele malen groter dan wat we kunnen verwerken. Het vinden van zinvolle informatie in deze overvloed aan data is daarom één van de grootste uitdagingen van de 21e eeuw. Zoekmachines zijn één van de mogelijkheden om grote dataverzamelingen te ontsluiten. Hun algoritmes hebben een immense ontwikkeling doorgemaakt. Waar zoekmachines eerst simpelweg zoektermen met documenten vergeleken, zijn het nu complexe systemen die vaak honderden signalen met elkaar combineren om zo de best mogelijke zoekresultaten voor elke gebruiker te genereren.

De huidige methoden voor het afstellen van zoekmachineparameters kunnen zeer effectief zijn, maar vergen vaak veel expertise en handmatige aanpassingen. Deze methoden zijn gebaseerd op zogenoemde *gecontroleerde* leertechnieken ("supervised learning"), wat betekent dat ze leren van handmatig geannoteerde voorbeelden van relevante documenten voor bepaalde zoekvragen. Goede handmatige voorbeelden zijn vaak niet of onvoldoende beschikbaar, zoals bij gepersonaliseerde zoektoepassingen, bij toegang tot gevoelige data en bij toepassingen die met de tijd veranderen.

Dit proefschrift richt zich op het ontwikkelen van nieuwe *online* leertechnieken, gebaseerd op het principe van versterkend leren ("reinforcement learning"). In tegenstelling tot gecontroleerde technieken kunnen deze direct van de interacties tussen zoekmachine en gebruiker leren. Deze interacties kunnen vaak eenvoudig verzameld worden, maar zijn door onzuiverheden en ruis moeilijk te interpreteren. De belangrijkste uitdaging is daarom het ontwikkelen van technieken die deze interactie goed kunnen interpreteren. De resultaten van dit proefschrift omvatten onder meer een zuivere stochastische methode die impliciete voorkeuren van gebruikers voor bepaalde zoekresultaten nauwkeurig kan detecteren en leermethodes die doeltreffend van de resulterende relatieve feedback kunnen leren.

De verworven analytische en empirische resultaten laten zien hoe zoekmachines effectief van gebruikersinteracties kunnen leren. In de toekomst kunnen deze en vergelijkbare technieken nieuwe manieren mogelijk maken om waardevolle informatie uit steeds grotere dataverzamelingen te ontsluiten.

Zusammenfassung

Die Menge digitaler Daten, die wir täglich produzieren, übersteigt unsere Möglichkeiten diese zu verarbeiten bei weitem. Eine der größten Herausforderungen des 21. Jahrhunderts ist es deshalb, nützliche Informationen in dieser Datenflut zu finden. Suchmaschinen bieten eine Möglichkeit, große Datensammlungen zu erschließen. Ihre Algorithmen haben eine immense Entwicklung durchlaufen. Aus Maschinen die Suchanfragen Wort für Wort mit Dokumenten vergleichen, sind komplexe Systeme geworden, die oft hunderte von Signalen kombinieren, um die bestmöglichen Suchergebnisse für jeden Nutzer zu generieren.

Heutige Methoden zur Optimierung von Suchmaschinen können sehr effektiv sein, benötigen aber meist ein großes Maß an Expertise und manuellem Aufwand. Sie basieren auf sogenannten *überwachten* Lernmethoden ("supervised learning"), die von manuell erstellten Beispielen relevanter Dokumenten für bestimmte Suchfragen lernen. Solche manuell erstellten Beispiele sind in vielen Bereichen nur bedingt verfügbar, zum Beispiel bei personalisierten Suchergebnissen, bei Zugang zu sensitiven Daten, oder in Suchanwendungen in denen die Nutzeranforderungen sich mit der Zeit verändern.

In dieser Dissertation werden neue *online* Lernmethoden entwickelt. Diese basieren auf dem Prinzip des verstärkenden Lernens ("reinforcement learning") und ermöglichen, im Gegensatz zu überwachten Lernmethoden, die Entwicklung von Suchmaschinen, die direkt von Interaktionen mit ihren Nutzern lernen. Spuren von Nutzerinteraktionen sind ein natürliches Nebenprodukt der normalen Nutzung von Suchmaschinen, und können deshalb die wirklichen Erwartungen von Nutzern widerspiegeln. Die wichtigste Herausforderung ist es jedoch, diese Interaktionen korrekt zu interpretieren, da sie durch Rauschen und Trends ("bias") beeinflusst werden. Die Beiträge dieser Dissertation umfassen unter anderem eine neue, wahrscheinlichkeitsbasierte, Methode, um Suchergebnisse durch Auswertung von Nutzerverhalten miteinander zu vergleichen. Darauf aufbauend werden online Lernmethoden entwickelt, die die resultierenden Vergleiche effektiv verarbeiten können.

Die erzielten analytischen und empirischen Ergebnisse zeigen wie Suchmaschinen effektiv von Nutzerinteraktionen lernen können. In Zukunft können diese und ähnliche Technologien neue Möglichkeiten eröffnen, um wertvolle Information aus stets größeren Datensammlungen zu erschließen.