Fast and reliable online learning to rank for information retrieval
Hofmann, K.

Citation for published version (APA):
Hofmann, K. (2013). Fast and reliable online learning to rank for information retrieval

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
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\textsuperscript{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 \textit{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-

\textsuperscript{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):

\[
NDCG = \sum_{i=1}^{len(l)} \frac{2^{rel(l[i])} - 1}{\log_2(i + 1)} \cdot \frac{i^{NDCG}}{i^{NDCG - 1}}.
\]

For a given result list \(l\) of length \(len(l)\), this metric sums over the gain that is based on the relevance label \(rel(l[i])\) 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 \(i^{NDCG}\) 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.

\[\text{Note that our formulation differs from earlier ones, including the one provided with the LETOR toolkit, where documents at rank 2 are not discounted (cf., (Järvelin and Kekäläinen, 2002)). Here, we use the formulation from (Burges et al., 2005) so that relevance differences at the highest ranks can be detected.}\]
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 \((x, y, q)\). Here, \(x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n\) 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 (\S2.3).

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
2. Background

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 $x$ for individual docu-
ments (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 exam-
ple, binary relevance scores (i.e., predicting whether a document is relevant or not) can
be learned using standard classification approaches (Nallapati, 2004), and regression ap-
proaches 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 exam-
ple, 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 $(x_{1,q}, x_{2,q}) \in \mathbb{R}^n \times \mathbb{R}^n$ (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 $x = x_{1,q} - x_{2,q}$. In this case, the loss function could be based on the clas-
sification 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 bot-
tom 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 av-
eraged 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 (Fünnkranz 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 $(x_{1,q}, \ldots, x_{m,q}) \in \mathbb{R}^{n \times m}$, 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 ($l_1(q), l_2(q)$). Here, rankers are deterministic functions that, given a query $q$, produce a ranked list of documents $d$. Given $l_1$ and $l_2$, interleaved comparison methods produce outcomes $o \in \{-1, 0, 1\}$ that indicate whether the quality of $l_1$ is judged to be lower, equal to, or higher than that of $l_2$, 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.

---

4If it is clear from the context which $q$ is referred to, we simplify our notation to $l_1$ and $l_2$. 
2. Background

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: $l_1, l_2$
2: $l = []; i_1 = 0; i_2 = 0$
3: $first_1 = random\_bit()$
4: while $(i_1 < \text{len}(l_1)) \land (i_2 < \text{len}(l_2))$ do
5:   if $(i_1 < i_2) \lor ((i_1 == i_2) \land (first_1 == 1))$ then
6:     if $l_1[i_1] \not\in l$ then
7:       append($l, l_1[i_1]$)
8:       $i_1 = i_1 + 1$
9:   else
10:     if $l_2[i_2] \not\in l$ then
11:       append($l, l_2[i_2]$)
12:       $i_2 = i_2 + 1$

// present $l$ to user and observe clicks $c$, then infer outcome (if at least one click was observed)
13: $d_{\text{max}} = \text{lowest-ranked clicked document in } l$
14: $v = \min \{ j : (d_{\text{max}} = l_1[j]) \lor (d_{\text{max}} = l_2[j]) \}$
15: $c_1 = \text{len}\{ i : c[i] = true \land l[i] \in l_1[1..v] \}$
16: $c_2 = \text{len}\{ i : c[i] = true \land l[i] \in l_2[1..v] \}$
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 $a$ (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**: \( l_1, l_2 \)
2: \( l = []; a = [] \)
3: **while** \( (\exists i : l_1[i] \notin 1) \lor (\exists i : l_2[i] \notin 1) \) **do**
4: \( \text{if } \text{count}(a, 1) < \text{count}(a, 2) \lor \text{rand\_bit}() == 1 \) **then**
5: \( j = \min \{ i : l_1[i] \notin 1 \} \)
6: \( \text{append}(1, l_1[j]) \)
7: \( \text{append}(a, 1) \)
8: **else**
9: \( j = \min \{ i : l_2[i] \notin 1 \} \)
10: \( \text{append}(1, l_2[j]) \)
11: \( \text{append}(a, 2) \)

// present 1 to user and observe clicks \( c \), then infer outcome
12: \( c_1 = \text{len} \{ i : c[i] = \text{true} \land a[i] == 1 \} \)
13: \( c_2 = \text{len} \{ i : c[i] = \text{true} \land a[i] == 2 \} \)
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).
2. Background

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

1: **Input:** $l_1, l_2$
2: $l = []$; $i_1 = 0$; $i_2 = 0$
3: $first_1 = random\_bit()$
4: while $(i_1 < len(l_1)) \wedge (i_2 < len(l_2))$ do
5:     if $(i_1 < i_2) \vee ((i_1 == i_2) \wedge (first_1 == 1))$ then
6:         if $l_1[i_1] \notin l$ then
7:             append($l, l_1[i_1]$)
8:             $i_1 = i_1 + 1$
9:         else
10:             if $l_2[i_2] \notin l$ then
11:                 append($l, l_2[i_2]$)
12:                 $i_2 = i_2 + 1$
13:     end if
14: end if
// present l to user and observe clicks c, then infer outcome
15: $v_1 = violated(l, c, l_1)$ // count constraints inferred from l and c that are violated by $l_1$
16: $v_2 = violated(l, c, l_2)$ // count constraints inferred from l and c that are violated by $l_2$
17: 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 $i$th 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
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:

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

When $\gamma = 0$, the agent cares only about maximizing immediate rewards through exploitation. As $\gamma$ 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;
2.4. Reinforcement Learning

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. 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

---

6In 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.
2. Background

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.\footnote{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).}

Most approaches for balancing exploration and exploitation in RL have been develop-
ed 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 \textit{\epsilon-greedy} exploration (Watkins,
1989), the agent selects an action with probability $\epsilon$ 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 perfor-
mance requires supplementing them with mechanisms for properly balancing exploration
and exploitation. To this end, exploration heuristics developed for value-function meth-
ods have been successfully adapted for policy search (Whiteson and Stone, 2006b).

Because balancing exploration and exploitation is considered important for optimiz-
ing 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 inter-
leaved 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 con-
text (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. So-
lutions 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
2.5. Baseline Online Learning to Rank Approaches

Through 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:

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

(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 $(x_1, \ldots, x_m)$.$^8$ Also, both learn a weight vector $w$ for linear-weighted combinations of these features. At any point $t$ during learning, a ranked list can be obtained from a current weight vector $w_t$ for a given query by scoring the candidate documents using $s = w_t^T \times (x_1, \ldots, x_m)$, 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 \( \hat{w} \) that minimizes the empirical loss \( L(w, x, y) \) given a set \( P \) of labeled training samples, each consisting of a feature vector \( x \) and a label \( y \):

\[
\hat{w} = \arg\min_w \left[ \frac{1}{|P|} \sum_{i=1}^{|P|} L(w, x_i, y_i) + \frac{\lambda}{2} ||w||^2 \right],
\]

where the last term is a regularization term. Using the hinge loss, i.e., \( L(w, x, y) = \max(0, 1 - yw^T x) \), 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 \( x = (x_{1,q} - x_{2,q}) \), where \( x_{1,q} \) and \( x_{2,q} \) 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 \( D \), learning rate \( \eta \), regularizer weight \( \lambda \), and an initial weight vector \( w_0 \). 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 \( d_i \in D \). The document feature vectors are then scored using the weight vector learned so far (\( w_{t-1} \)), and sorted by this score to obtain an exploitive result list (the best ranking given what has been learned so far).

The constructed exploitive result list is shown to the user, and clicks on any of the result documents are observed. From the observed clicks \( c \), all possible labeled document pairs \( P \) are inferred using the pairwise labeling method described above (Joachims, 2002). The labeled samples in \( P \) are then used to update the weight vector \( w \). 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 \( w_t = w_{t-1} + \eta y(x_1 - x_2) - \eta \lambda w_{t-1} \). Here, we use the unregularized version of this update rule (by setting \( \lambda = 0 \)) and use a small constant \( \eta \). This formulation was found

\(^9\)Provided online at http://code.google.com/p/sofia-ml/.
2.5. Baseline Online Learning to Rank Approaches

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

1: **Input**: \( \mathcal{D}, \eta, \lambda, w_0 \)
2: **for** query \( q_t (t = 1..\infty) \) **do**
3: \( X = \phi(\mathcal{D}|q_t) \) // extract features
4: \( s = w_T^T X \)
5: \( l = \text{sort}\_\text{descending}\_\text{by}\_\text{score}(\mathcal{D}, s)[1:10] \)
6: Display \( l \) and observe clicked elements \( c \).
7: Construct all labeled pairs \( \mathcal{P} = (x_1, x_2, y) \) for \( q_t \) from \( l \) and \( c \).
8: **for** \( (x_1, x_2, y) \) in \( \mathcal{P} \) **do**
9: \( \text{if } y w_t^T (x_1 - x_2) < 1.0 \) and \( y \neq 0.0 \) **then**
10: \( w_t = w_{t-1} + \eta y(x_1 - x_2) - \eta \lambda w_{t-1} \) // update \( w_t \)

...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 \( f(l_1, l_2) \), which compares two result lists \( l_1 \) and \( l_2 \) using user clicks \( c \) (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, \( g(\delta, w) \) is provided to generate candidate rankers. The remaining inputs are the step sizes \( \alpha \) and \( \delta \), and an initial weight vector \( w_0 \). An optional parameter \( \theta \) 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 \( w_t \). When a query \( q_t \) is observed, a new candidate weight vector \( w'_t \) is generated using \( g(\cdot) \) (line 4). Then, result lists for \( q_t \) are generated using both the current best (\( w_t \)) and the candidate (\( w'_t \)) weight vector (line 5; \( \text{generate}\_\text{list}(\cdot) \) 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 \( f(l_1, l_2) \) (line 6). If \( w'_t \) wins the comparison, \( w_t \) is updated using the update rule \( w_t \leftarrow w_t + \alpha u_t \) (line 8). Otherwise, \( w_t \) is not changed. Lines 11–14 of Algorithm 5 shows how historical...
Algorithm 5 Baseline online learning to rank algorithm for the listwise setting, based on (Yue and Joachims, 2009).

1: **Input**: $f(l_1, l_2), g(\delta, w), \alpha, \delta, w_0, \theta$ (default: 0)
2: $h \leftarrow []$
3: for query $q_t$ ($t \leftarrow 1..\infty$) do
4:   $(w'_t, u_t) \leftarrow g(\delta, w_t)$ // generate candidate ranker
5:   $l_1 = generate\_list(w_t); l_2 = generate\_list(w'_t)$
6:   $(o_L, l, a, c) \leftarrow f(l_1, l_2)$
7:   if $o_L > 0$ then
8:      $w_{t+1} \leftarrow w_t + \alpha u_t$ // update current best ranker
9:   else
10:      $w_{t+1} \leftarrow w_t$
11:   // maintain historical data if needed
12:   if $\theta > 0$ then
13:      if $\text{len}(h) = \theta$ then
14:         $\text{remove}(h, h[0])$
15:      $\text{append}(h, (l, a, c))$

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

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

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, $\text{generate\_candidate}(\cdot)$ is used to generate candidate weight vectors as follows (Algorithm 6). First, a vector $u$ is generated by randomly sampling a unit vector. Then, $w'$ is obtained by moving $w$ by a step of size $\delta$ in the direction $u$. 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.