Probabilistic Modeling in Dynamic Information Retrieval

Marc Sloan

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

December 12, 2015

I, Marc Sloan, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Dynamic modeling is used to design systems that are adaptive to their changing environment and is currently poorly understood in information retrieval systems. Common elements in the information retrieval methodology, such as documents, relevance, users and tasks, are dynamic entities that may evolve over the course of several interactions, which is increasingly captured in search log datasets. Conventional frameworks and models in information retrieval treat these elements as static, or only consider local interactivity, without consideration for the optimisation of all potential interactions. Further to this, advances in information retrieval interface, contextual personalization and ad display demand models that can intelligently react to users over time.

This thesis proposes a new area of information retrieval research called Dynamic Information Retrieval. The term *dynamics* is defined and what it means within the context of information retrieval. Three examples of current areas of research in information retrieval which can be described as dynamic are covered: multi-page search, online learning to rank and session search. A probabilistic model for dynamic information retrieval is introduced and analysed, and applied in practical algorithms throughout.

This framework is based on the partially observable Markov decision process model, and solved using dynamic programming and the Bellman equation. Comparisons are made against well-established techniques that show improvements in ranking quality and in particular, document diversification. The limitations of this approach are explored and appropriate approximation techniques are investigated, resulting in the development of an efficient multi-armed bandit based ranking algo-

rithm. Finally, the extraction of dynamic behaviour from search logs is also demonstrated as an application, showing that dynamic information retrieval modeling is an effective and versatile tool in state of the art information retrieval research.

Acknowledgements

This thesis would not exist without the guidance, intelligence and support of my supervisor and friend, Dr. Jun Wang. He has been an expert in knowing when to step in to help me move in the right direction, and when to stand back and let me make my own way. His advice and actions have shaped my life for the past 5 years, and for that I owe him so much.

Dr. Grace Hui Yang has been a close collaborator and in all but name, a second supervisor to me. Our numerous discussions and the work we've accomplished together has massively impacted this thesis, and I look forward to the continuation of our partnership. I also would like to thank my actual second supervisor Dr. David Silver for his expertise on reinforcement learning and our discussions throughout my PhD.

The MSR CLUES team, in particular Paul Bennett, Kevyn Collins-Thompson and Milad Shokouhi, have also been hugely influential. My internship with them was an unforgettable experience and the time that they invested in me has made me a significantly better researcher.

My PhD would not have been possible without the financial and pastoral support given to me by Prof. Philip Treleaven and Yonita Carter at the *UK PhD Centre in Financial Computing*. I am enormously grateful that they accepted me onto their program and kick-started my academic development.

My research group have been a constant source of help, camaraderie and friendship over the years. In particular, the old crowd of Dr. Shuai Yuan, Dr. Bowei Chen, Dr. Tamas Jambor and Dr. Jagadeesh Gorla, and the new crowd Manisha Gupta, Rishabh Mehrotra, Weinan Zhang and Xiaoxue Zhao, all of whom I know

will have great success in their lives.

My friends too have been unfailing in their support and encouragement, despite me being the perennial student who can never afford to buy a round of drinks. So to John Kennedy, Mark Curry, Gareth Hayman, Paul King, Andrew Crozier, Enrico Fantoni, Kjerstin Østenseth and also the folks from the IAESTE London LC, the next round is on me.

Finally, I would not be at this position in my life without the love from my family. My brothers, whose ambition and success have spurred me on further than I could have managed alone. My father, who always knew I would go on to do something like this and would have loved to have read this thesis, I miss him. And my mother, who had to be two parents for three young boys and never gave up. All of my accomplishments are because of her.

And Sarah. For tolerating the late nights, the stresses, the travelling and long-distance calls. Who proof-read this thesis, listened to my ideas and watched me practice presentations. Who challenged me and advised me. Who loved me throughout and put a smile on my face when I needed it most. Thank you.

I dedicate this thesis to my late father, who showed me the way to go,
and to my mother, for helping me get there.

Contents

List of Figures

List of Tables

Notation

a Action.

$\mathscr{A}$ Action space.

b Rank position bias.

C Click variable.

d Document.

D Corpus.

f^R Reward function.

g Mixture model parameter.

G Mixture model random variable.

i Rank position.

M Number of ranked documents in an SERP.

o Observation.

O Space of observations.

q Query.

r Relevance score.

R Relevance variable.

s State.

$\mathscr{S}$ Set of states.

t Stage or time step.

T Time horizon.

U_D Dynamic utility function.

U_S Static utility function.

Z Sample size.

Θ Observation probability function.

λ Exploration parameter.

Λ Index score.

σ Standard deviation.

Σ Covariance.

τ Relevance update function.

ω Path discount function.

Chapter 1

Introduction

> We shall attempt to define intelligence, as have others before us, as *"goal-directed adaptive behaviour"*.
>
> Robert J Sternberg
>
> *Handbook of human intelligence* (1982, pg. 3)

The **Information Retrieval (IR)** ecosystem is a dynamic one: users translate their information needs into an assortment of meaningful interactions with IR systems; corpora and **search logs** follow behind the slowly shifting world they are gathered from; language, interpretation and intent all change with time. **Dynamic Information Retrieval (*DIR*)** is a framework for modeling and understanding IR systems within this ecosystem. These dynamics are evident in many existing IR systems and data collections, yet are not fully utilised by conventional IR methods. In this thesis, the difference between dynamic and conventional IR is explored, the elements that constitute a framework for general dynamic IR systems are identified and then applied to research in current areas of academic interest in IR. Along the way, key characteristics of *DIR* such as diversity and exploration are covered in more detail.

In conventional IR, probabilistic modeling has largely been confined to what can be defined as *static* problems, where a set of parameters is learned from a dataset and fixed for use in an IR system. Examples include ad hoc ranking and retrieval relevance scoring such as the **BM25** model [2], topic modeling using latent Dirichlet allocation [3] or learning to rank [4]. Underlying the solutions to these problems

are static frameworks such as the classic **Probability Ranking Principle (*PRP*)** [5] which justifies the simplest and most powerful ranking rule in IR: ranking documents in decreasing order of their probability of relevance.

The described models are not capable of representing search tasks that operate over multiple stages nor can they incorporate user feedback. Change is at the heart of a modern information retrieval system. Search tasks are complex and often exploratory, with the user broadcasting signals of search intent over multiple stages of retrieval, specializing or generalizing their information needs over time [6]. Further to this, advances in IR interface, personalization and ad display demand models that can react to users in real time and in an intelligent, contextual way. Examples include query reformulation in session search [7], item ratings in collaborative filtering [8] and maximising revenue generated from web advertisements [9]. The aim of the proposed dynamic information retrieval modeling is to find IR solutions that are responsive to a changing environment, that learn from past interactions and that predict future utility.

The term '*dynamics*' is taken from physics to describe systems that manipulate the Newtonian laws of motion, for example, by altering the velocity of a pendulum using damping forces [10]. Control theory, a related subfield, is the mathematics of closed feedback looping systems that maintain some form of equilibrium or **state**, for instance, a pendulum that continuously oscillates due to the repeated application of some force [11]. Such systems are described as dynamic **agents**, comprised of a **feedback** sensor that measures a **utility** value for the environment, which is used by the agent to determine an **action** that affects the environment. A dynamic agent for IR is depicted in Fig. (1.1).

Agents exhibit *goal-directed adaptive behaviour*, a phrase used by Robert J Sternberg to describe human intelligence [12]. While he also merits memory, reasoning and abstraction amongst other definitive traits, nonetheless one can label a dynamic agent an intelligent one, responsive to the dynamics of its real world setting so that it can achieve its goal. Dynamic agents are resistant to adverse change or error and are able to learn and adapt to their surroundings. Many agents maintain a

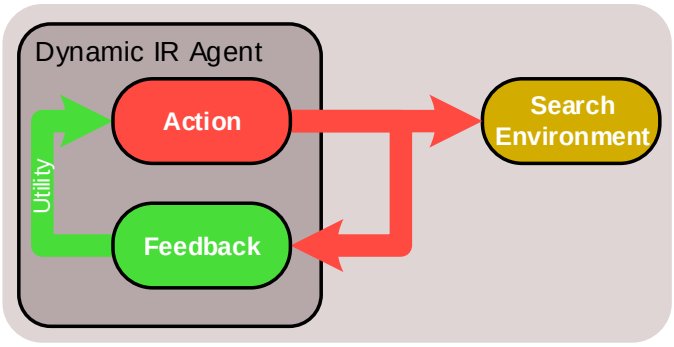


Figure 1.1: The closed feedback loop of a dynamic agent in IR. The agent senses its search environment using its feedback sensor and converts this into a utility value. This value determines the action that the agent takes, which affects the external environment, and the process repeats.

model of their environment so that they can understand how it will react given some input into it. This is encapsulated by the three characteristics common to dynamic agents:

1. The ability of the agent to perceive its environment through feedback.

2. The ability to respond to this feedback, to adapt and change its actions accordingly.

3. A long-term goal or utility towards which the agent is driven.

In *DIR*, these characteristics can be attributed to user feedback, temporal dependency and an overall utility value [13].

The objective of this thesis is to provide a comprehensive introduction to the field of dynamic information retrieval. This is achieved by defining a framework for dynamic IR as a natural progression in IR research complexity: where early research concerned *static* problems such as ad hoc retrieval, which gave way to *interactive* tasks such as those incorporating relevance feedback, which has lead to *dynamic* systems for tasks such as session search ranking. The framework incorporates dynamic elements common to *DIR* systems, and these are explored through some its various applications: multi-page search, online learning to rank and session search. Practical dynamic algorithms build on the work in reinforcement learning and artificial intelligence by making use of dynamic programming and multi-armed bandit

theory. As a result, interesting insights are found into how to optimally explore rankings over time, document diversification and user behaviour.

1.1 IR Systems

IR systems can be conceptualized into three categories: static, interactive and dynamic, each a generalization of the previous. Similar trends are observed in other areas of IR research, for example the early term-based vector space retrieval model gave way to more complex models like BM25 and language modeling [14]. Likewise, the evolution from static to interactive and then dynamic IR reflects the increasing complexity of search problems and the need for responsive solutions.

1.1.1 Static IR

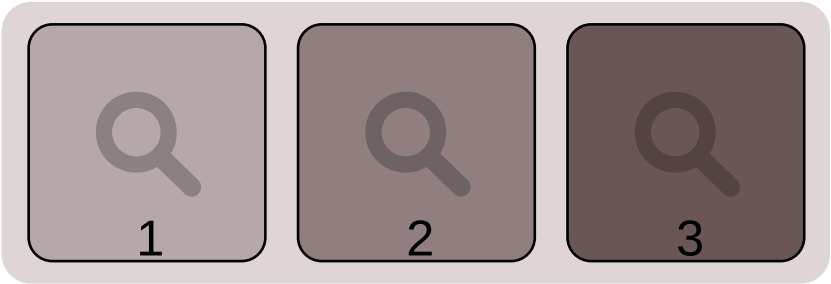


Figure 1.2: The independent stages of static IR.

Static IR encompasses problems in information retrieval that are resolved in a single time step or interaction, or multiple independent interactions (represented in Fig. (1.2)), not requiring consideration for how the state of the system has changed following the interaction. Many traditional areas in IR can be described as static, for instance, ad hoc ranking and retrieval where document relevance scores for query terms are typically generated in advance of retrieval and fixed. These scores and the rankings they give rise to are independent of the user's preceding or future actions or the state of the system. Re-calculating the scores based on some real-time user input would be a slow and expensive operation.

A static IR system is illustrated in Fig. (1.3). Here, a retrieval system is generating two pages of search results for the query jaguar. This ambiguous query gives results for webpages belonging to the subtopics `car`, `animal` and `guitar`. These

results are diversified across both pages in the static system so that the results can cater to the widest range of users and their search intent. In this static system, the results on page 2 are unaffected by any interaction that occurs on page 1, a different user searching for the same query would encounter the same results.

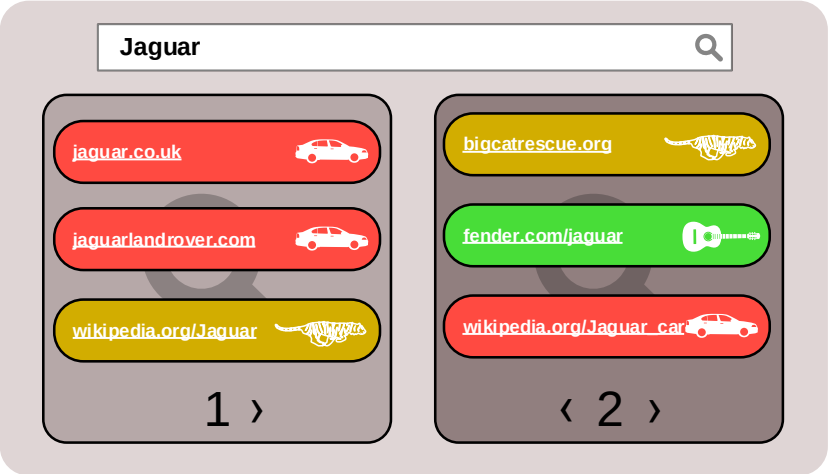


Figure 1.3: Two pages of static search results for the query `jaguar`, categorized by subtopic.

There are many different ways to perform ranking and retrieval in this environment. A simple, traditional method is the vector space model [15], which treats queries and documents as vectors and ranks documents in decreasing order of their distance from the query. One could argue that a system such as this is a dynamic agent as it senses the user's information need via their query, it responds by generating a ranked list of documents and its goal is to satisfy the information need. In this scenario there is only one time step whereas dynamic systems operate over many time steps. The scenario could be extended by allowing users to search for multiple queries consecutively in a session. Still, traditional ranking and retrieval models would return the same results regardless of the order of the queries or the behaviour of the user and the search engine. In this thesis, such models are not defined as dynamic and instead they are labelled as *static*.

The search scenario given in the example above could be improved by using relevance feedback to personalize the results on the second page.

1.1.2 Interactive IR

An **interactive IR** system is one that extends a static system by incorporating user feedback. In interactive IR, each stage is dependent on the previous stage (represented in Fig. (1.4)).

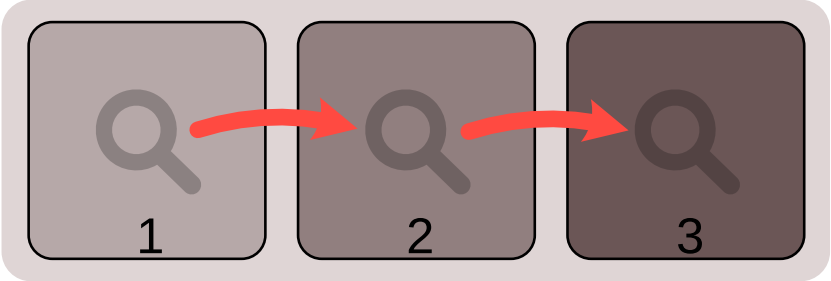


Figure 1.4: Dependent stages in interactive IR.

Once feedback from the user has been observed, an interactive IR system can then improve the search experience. For example, the well-known Rocchio algorithm [16] uses explicit feedback to improve the user's query. Recommender systems, ad selection and query auto completion are examples of modern systems that incorporate feedback to improve performance.

Interactive IR is a progression from static IR in that it deals with the complexity of user interaction by operating over multiple stages, making use of user feedback (such as document clicks). The stages may represent multiple queries in a search session, multiple sessions in a user's search history, different users in a search log and so on. An interactive system may begin by using a static method but will then continue to adapt to the user after each stage.

When an interactive technique such as the Rocchio algorithm is applied to the static ranking example in Fig. (1.3), the second page of search results can be personalized based on the user's subtopic preference from page 1. For instance, if a user clicked on a `car` related webpage on page 1, then the second page of results could be updated as shown in Fig. (1.5). Likewise, if their click preference were instead for the subtopic `animal`, then the algorithm would respond appropriately to give an alternative ranking for page 2. This is a user targeted improvement over the static ranker's second page which continued to display a mix of subtopics.

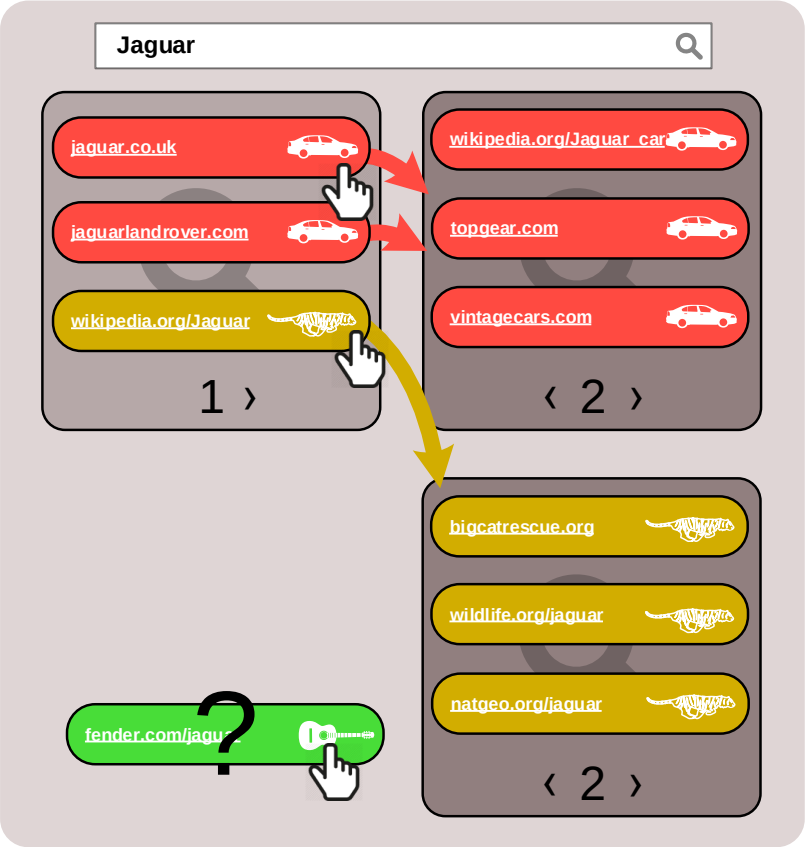


Figure 1.5: Clicked webpages lead to the personalization of the second page of results based on the subtopic clicked on page 1 of the ranking in Fig. (1.3), but subtopic `guitar` is no longer represented.

A problem with this interactive approach is that it does not allow for the personalization of results for those users not interested in the two subtopics found on the first page. For instance, a `guitar` related webpage is contained in the second page of the static IR example, but users interested in this subtopic are not given the opportunity to provide the necessary feedback for the search system to learn their preference. A dynamic IR system can help resolve this situation.

1.1.3 Dynamic IR

The stages in **Dynamic IR** are defined as being dependent on both past and predicted future interactions (represented in Fig. (1.6)). A *DIR* system responds to the dynamics of its real world setting so that it can achieve its goal. Such systems are resistant to adverse change or error and are able to learn and adapt.

Dynamic IR is a natural evolution of the described static and interactive mod-

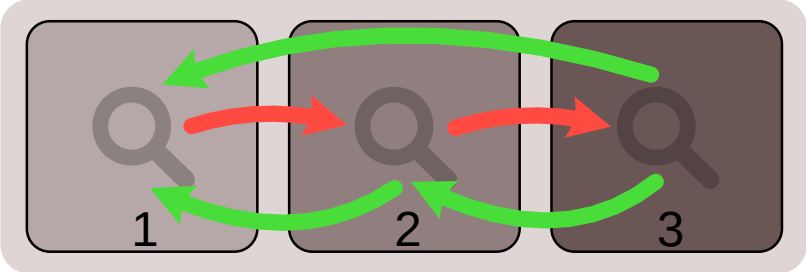


Figure 1.6: Forward and backwards dependency in dynamic IR.

els. As is the case with an interactive system, a dynamic system may collect feedback from a static system and respond accordingly. A key difference is how the objective of the system is optimised and defined; in interactive retrieval only immediate rewards are considered, whereas in dynamic retrieval the overall reward is prioritised. As a result, the action chosen at each time step in a dynamic system is made in consideration of all observed past and potential future interactions.

In the interactive IR example in Fig. (1.5), the results on the first page followed those of the static ranking and the second page was enhanced by interactively incorporating user clicks, but at the cost of alienating users with a preference for subtopic `guitar`. This can be resolved using a dynamic approach that determines optimal rankings for both pages of search results, such as the multi-page search algorithm covered in more detail in Chapter 4. Fig. (1.7) shows that a first page of results can be found that balances the learning of a user's preference and the display of relevant documents.

When this approach is used, it is found that a diversified first page ranking maximises the learning potential of the system, so that improved, targeted results can be returned for the next page. This can also be reiterated as the explore/exploit property common in reinforcement learning, where on the first page the documents are explored, then on the second page relevant documents are exploited. Any performance losses suffered due to the first page diversification are balanced by gains made in the second page.

In summary, the differences between the three types of IR system in this conceptual model are: static systems are those that operate over a single stage or oth-

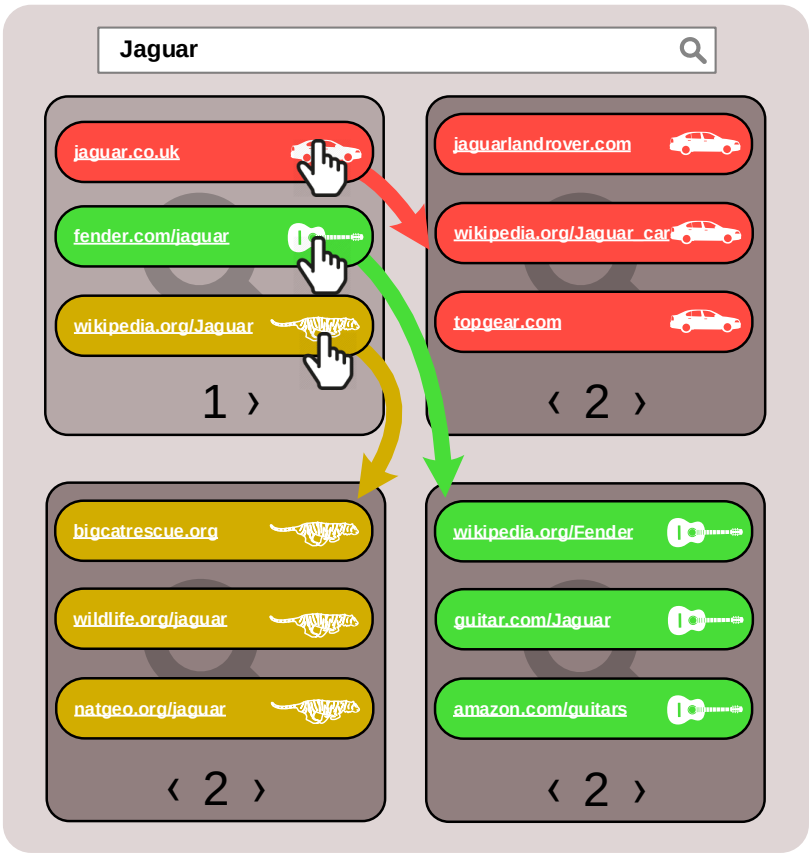


Figure 1.7: The first page ranking has been diversified so that the search system is better able to learn the user's second page preference, improving the overall search experience for all users.

erwise multiple stages which are independent of one another. Interactive systems extend static systems by introducing local dependency from one stage to the next and optimizing for individual goals per stage. A dynamic system extends an interactive system by focusing on a single goal that forces dependency across all stages.

A dynamic system is one which changes or adapts over time and has a range of applications, from a ranking and retrieval algorithm, to an advert recommender or a query suggestion model. In this thesis, dynamic IR is defined as the modeling of adaptive, responsive, goal-oriented information retrieval systems. A general IR framework for both static and interactive IR is defined, and this is used to motivate a model for dynamic IR problems. This framework then underpins the remainder of the research in this thesis.

1.2 Multi-Page Search

Multi-Page Search (MPS) is an application of dynamic IR explored extensively in this thesis. The scenario concerns the ranking of documents over multiple pages of search results, where documents are retrieved for a single query, ranked and then segregated into pages of documents. On each page, a user may examine and click on documents and the goal is to use this feedback to improve the rankings over all pages of search.

This scenario is used in the examples in Fig. (1.3), Fig. (1.5) and Fig. (1.7). As demonstrated in the examples, the multi-page search problem is familiar and easily definable in static, interactive and dynamic frameworks and intuitively expresses their similarities and differences. The derivation and comparison of practical applications of MPS solutions using all three frameworks is a key contribution of this thesis.

The multi-page search scenario concerns a single query and a single set of documents, making it a simple problem formulation for preliminary investigations into the efficacy of the *DIR* framework. Further to this, existing methods from IR research can be readily used for each of the elements in the framework for the MPS scenario, and are also applicable in the static and interactive cases. Finally, the **TREC (Text REtrieval Conference)** data collections and their corresponding relevance judgements are openly available and ideally suited to evaluation in this area.

In this thesis, practical applications of the static, interactive and dynamic frameworks to the MPS scenario are directly compared. Through TREC experiments it is found that the application of the *DIR* framework can lead to the diversification of search results on the first page, as was the case in the examples in the previous section. This is a result of the natural exploration that occurs in dynamic systems. The MPS experiments are taken further in a thorough investigation of a specific application of the *DIR* framework that demonstrates comparable performance to established IR techniques such as the Rocchio algorithm and Maximal Marginal Relevance [17].

1.3 Online Learning to Rank

Online learning to rank is defined 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"* [18]. It is a natural *DIR* problem and a further application of the dynamic IR framework in this thesis.

The field was first motivated as an extension of the **learning to rank** framework, the application of supervised machine learning algorithms to the space of search tasks. As with other areas of IR, the goal in learning to rank is to find an optimal ranking of documents for an information need. In this case, document relevance labels (generated by assessors or otherwise) are used to train a classifier (such as an SVM) to identify relevant documents, or optimal rankings of documents, for validation in a test set. A range of document, query, session and user features are typically used to train the classifier [4]. The learning to rank classification of relevance labels or regression of ranking scores fall broadly into three categories:

Pointwise The binary relevance label of individual documents are learned using a classifier [19].

Pairwise Instead of learning a direct relevance label, the correct order of pairs of documents is learned, as is the case with the well-known RankSVM algorithm [20].

Listwise IR measures such as **Normalised Discounted Cumulative Gain (nDCG)** or **Mean Average Precision (MAP)** are directly optimised over lists of documents, as in the widely used LambdaMART algorithm [21].

A barrier to effective learning to rank is the acquisition of relevance labels, many of which are needed to train accurate, reliable and generalizable ranking systems. A common solution is to use explicit labels provided by users, often to their chagrin, or else noisy, implicit labels such as clicks found in abundance in search logs. In whatever form relevance labels are obtained, they come at some cost and also vary in their informativeness. Active and semi-supervised learning techniques

can therefore be employed to minimise such costs and maximise the learning potential of training sets, perhaps dynamically, over time. Once trained, a ranking classifier can be difficult to maintain, particularly in light of shifting search intents and new documents. Online learning to rank has been developed as a solution to these problems.

In online learning to rank, feedback signals from the user are used as relevance labels and used to update the ranking algorithm when received. The search ranking may change over time in response to this feedback, gradually converging on an optimal ranking that is informed by user input. Such systems can generate results targeted to users that can adapt over time and minimise the overheads of expensive label collecting. Examples of online learning to rank applications include ranking and retrieval [22], document diversification [23], evaluation [24], news article recommendation [25] and ad display [26].

It is clear that online learning to rank is the application of dynamic IR to the search ranking problem. In this case, the user at each time step is dynamic, in that online learning to rank occurs over a population of users. The overall objective is to learn the relevance of a set of documents or learn a ranking that optimises a given evaluation metric. The feedback signals are the user clicks, ratings, comparisons or any other sign of relevance obtained after each search.

Reinforcement learning techniques such as the multi-armed bandit are used in online learning to rank and introduce concepts such as exploration, where potentially relevant documents are displayed to the user, and exploitation, where documents known to be relevant are displayed. The application of reinforcement learning algorithms to *DIR* problems is one method of reducing issues with computational complexity, a limitation of the *DIR* framework. Other approximation methods are investigated in this thesis, such as Monte Carlo sampling and the sequential ranking decision in both the MPS and online learning to rank scenarios.

1.4 Session Search

A **search session** is defined as a period of time over which a user engages with a search engine by issuing more than one query in pursuit of satisfying an information need [27]. Often in practice, a boundary of 30 minutes is used to distinguish different session periods [28]. Queries in session search differ from ad-hoc queries in that they are more likely to be ambiguous due to the user being unsure how to explicitly define their information need [29] or explorative when the user is actively seeking a broad range of information on a subject [30].

Session search is a dynamic procedure with many user interactions. Like with MPS, a user issues a query and the search system retrieves a ranked list of relevant documents and the user may examine documents and click on webpages. The difference is that the next interaction occurs when the user *reformulates* their query, retrieving a new ranked list of documents. A search session consists of several **query reformulations** that reflect the user's shifting cognitive focus and understanding of their information need, and stops when they are satisfied or become frustrated or bored [31]. An example session search from the TREC 2013 session track dataset is given in Table (1.1).

Understanding the dynamic user behaviour that occurs during session search has been a topic of interest in IR research in recent years. The intentions of users during exploratory searches [32], learning when session search personalization is useful [33] and how the relevance of previous documents affects ranking quality [34] are all examples of current research in the area. In a search session, the user explores the information space by inputting queries, examining retrieved documents and clicking on those that seem relevant. In this thesis, TREC session track search logs are studied in order to extract meaningful interaction data across queries in a session. From the *DIR* framework observable user actions are defined, and following a thorough analysis, conclusions about the dynamic behaviour of users in session search can be inferred.

Table 1.1: Session 87 from the TREC 2013 Session Track dataset [1], the topic is "*Suppose you're planning a trip to the United States. You will be there for a month and able to travel within a 150-mile radius of your destination. With that constraint, what are the best cities to consider as possible destinations?*".

#	Query
1	best us destinations
2	distance new york boston
3	maps.bing.com
4	maps
5	bing maps
6	hartford tourism
7	bing maps
8	hartford visitors
9	hartford connecticut tourism
10	hartford boston travel
11	boston tourism
12	nyc tourism
13	philadelphia nyc distance
14	bing maps
15	philadelphia washington dc distance
16	bing maps
17	philadelphia tourism
18	washington dc tourism
19	philadelphia nyc travel
20	philadelphia nyc train
21	philadelphia nyc bus

1.5 Research Questions

In sum, the research questions that this thesis sets out to resolve are the following:

RQ1 *What is dynamic IR and can a framework for it be defined in the context of existing IR research?*

RQ2 *Can the* DIR *framework be used in a practical setting? What are the benefits/drawbacks to its use? How does its use differ from other IR frameworks?*

RQ3 *In what ways can reinforcement learning methods, sampling and other approximations be used to minimise the impact of complexity in the* DIR *framework?*

RQ4 *How should IR problems in the* DIR *framework be evaluated? What are the*

problems encountered when evaluating DIR *algorithms?*

RQ5 *What does the* DIR *framework reveal about the interactions that occur in a dynamic system? Can search logs from a dynamic system be used to understand dynamic interactions?*

1.6 Thesis Structure

An overview of the background work relevant throughout this thesis is provided in Chapter 2. It covers topics in IR such as click modeling and diversification, as well as the use of economic ideas such as utility and portfolio theory. Following this is a discussion of the related work on dynamics in IR and key algorithms and technologies in reinforcement learning used in this thesis.

The central argument of this thesis is presented in Chapter 3, which explicitly defines the static and interactive IR frameworks and uses them to motivate the dynamic IR framework (**RQ1**). After some analysis of its components, an experiment in the multi-page search scenario is presented comparing the different frameworks and demonstrating diversification as an effect of applying the *DIR* model (**RQ2**).

Chapter 4 continues the work on MPS by looking more explicitly at the scenario and defining a practical algorithm for the *DIR* framework (**RQ3**). Experiments are then conducted using TREC data and compared against other ranking algorithms under a number of different settings, demonstrating the effectiveness of the technique (**RQ2**).

The online learning to rank problem is tackled in Chapter 5, where the *DIR* framework is combined with a multi-armed bandit algorithm to create a simple ranking rule for learning optimal and diverse rankings over time (**RQ3**). Experiments on both simulations and search log data verify the effectiveness of the approach and also serve to highlight the difficulties in evaluating *DIR* algorithms (**RQ4**).

The subject of dynamics found in search logs is explored further in Chapter 6, where dynamic user behaviour is extracted from session search logs. By identifying features from the logs as components in the *DIR* framework, insights can be found as to how the **Search Engine Results Page (SERP)** affects query reformulation in

session search (**RQ5**).

Following on from this, the thesis concludes in Chapter 7, along with a glossary of important terms and related publications in the appendices.

Chapter 2

Background

This chapter serves to provide a cogent background for the understanding of research topics relevant to all of the chapters in this thesis. The related literature to the topic of each of the chapters is included within the chapter itself.

The background literature can be conveniently split into two sections. In the first, the broad field of information retrieval and the subtopics relevant to this thesis are covered. This includes general work on ranking and retrieval [35], diversification [36], user feedback [37] and click modeling [38]. The role of economics in IR is also touched upon, where utility theory [39] and document diversification using portfolio theory [40] are discussed.

The background on dynamics itself and its application to IR is explored in the second section. Starting with relevance feedback and the Rocchio algorithm [16], specific areas of research in reinforcement learning are then covered, including the Markov decision process [41], its partially observable variant and their solutions using the Bellman equation and dynamic programming [42]. The chapter finishes with an overview of the multi-armed bandit literature [43].

2.1 Information Retrieval

2.1.1 Ranking and Retrieval

The most prominent problem in information retrieval research is the retrieval of **documents** that are relevant to a specific user need (or **query**) and the ranking of those documents. A common way of determining if a document is relevant to an

information need is to calculate its **probability of relevance**, the posterior probability that, given the query issued by the **user**, the document is relevant [35].

Initially, IR systems labelled documents as relevant or non-relevant under the boolean model. Relevance models of increased complexity have since been developed, such as the Robertson-Spärck Jones probabilistic models [44], BM25 weighting [2], language modeling approaches [45] and the recent developments in learning to rank [20]. Each of these methods finds a value with which to score the **relevance** of a document to a query.

The scoring of document relevance can be affected by the nature of the query itself. Queries can be classified into the following three types [46]:

Navigational A query for which a user has a specific website or domain in mind, for example, the query `bing maps` in Table (1.1). Typical search log characteristics of such queries are the observance of single clicks on highly ranked documents.

Informational/Explorative A query that represents an ill-defined or broad information need, for example, the query `boston tourism` in Table (1.1). Users issuing these types of queries may search for multiple queries within a session and click on many documents, changing their query as they learn more about their subject of interest.

Transactional A query related to a purchase or transaction that the user wishes to make, for example, the query `philadelphia nyc bus` in Table (1.1). These queries can lead to explorative search behaviour, but with a tightly focused objective.

2.1.2 Clickthroughs

When evaluating ranking and retrieval systems, it can be difficult to measure their quality without direct access to the users of the system. One solution is the use of standardised TREC data collections as a source of high quality relevance judgements, corpora and text queries, though they are limited in their academic scope.

Another solution is to collect explicit relevance feedback from users but this is expensive and difficult to obtain.

Instead, clicked search results (or **clicks/clickthroughs**) are a commonly used way to infer user satisfaction; the observation of a user clicking on a document is a signal that the document is likely to be relevant to the query [47]. Studies have demonstrated the positive correlation between clicks and document relevance [48] and labels from explicit relevance feedback [49]. Clickthroughs can be easily recorded through the search engine server and are unobtrusive and cheap to collect in large quantities.

Nonetheless, clicks can be unreliable in their representativeness of a user's search interest, being both noisy and the subject of presentation bias. In a study where search system quality was systematically reduced, it was found that click behaviour varied more due to different users and search topics than due to the system quality itself [50]. Another study found that a search system performed poorly when it re-ordered search results based solely on clickthrough data [51].

The rank position of a search result has been found to be one of the strongest causes of bias in clicks [52], which has been verified through extensive eye-tracking studies [53]. This bias is a consequence of the *trust* that a user places in the ability of a search engine to correctly find and rank relevant documents. A user typically reads a list of search results from top to bottom with decreasing attentiveness, making them considerably more likely to view and click on a top ranked document, even if lower ranked documents are actually more relevant. SERP features such as URL length and caption readability [54], or session level features such as the impression position and the average number of documents examined [55] have also been shown to affect the probability that a user clicks on a document.

2.1.2.1 Click Models

When using clicks from search logs to help build and improve search systems, rank bias can be taken into account by modeling the clicks using a number of **click models**. These are probabilistic models that represent a user's typical behaviour with an SERP, typically modeling the likelihood that a user will examine a snippet

or click on a document. Three simple and well-known models are the *examination hypothesis*, *cascade* [38] and *dependent* click models [56].

In the examination hypothesis model, the probability of a click C given a document d ranked at position i is given by

$$P(C|d,i) = P(R|d) \times b_i \tag{2.1}$$

where R is the hidden relevance of d (typically a binary label of relevant or not relevant) and $b_i \in [0,1]$ is a parameter representing the *bias* of the ranking position.

In the cascade model,

$$P(C|d,i) = P(R|d) \prod_{j=1}^{i-1} \big(1 - P(R|d_j)\big) \tag{2.2}$$

where d_j is the document ranked at position j. Here, the probability of a click depends on the relevance of documents ranked at higher positions, modeling a user who examines documents in a list from top to bottom and leaves the search results page after finding a relevant document. An observation of a click means that all documents ranked above must have been skipped by the user.

In the dependent click model,

$$P(C|d,i) = P(R|d) \prod_{j=1}^{i-1} \big(\underbrace{1 - P(R|d_j)}_{\text{Not relevant}} + b_j \underbrace{P(R|d_j)}_{\text{Relevant}}\big) \tag{2.3}$$

which extends the cascade model to include the effect of rank bias on the probability of clicks on higher ranked documents.

Click models are an active area of research, with recently developed models incorporating sequential decision making through Markov chains [57], dynamic Bayesian networks [58] and, related to the work in this thesis, a POMDP to model more complex user interactions [59].

2.1.3 Document Diversification

Queries expressed in natural language can be ambiguous, particularly when containing homographic terms. For instance, the query term 'jaguar' (as used in the examples in Fig. (1.3), Fig. (1.5) and Fig. (1.7)) can refer to webpages representing subtopics `animal`, `car` and `guitar`. A conventional ranking algorithm that makes use of the *PRP* may not take this into account and only display results for one or two of the subtopics, ignoring users interested in the third. To present relevant results for all users, documents from all subtopics should be displayed, even if they do not score highly enough to be ranked by the *PRP*. In this scenario, the *PRP* no longer ranks documents optimally due to the dependencies that exist between documents [60]. This is explored more explicitly in Chapter 3.

Diversification in ranking and retrieval is an active area of research. An early technique known as **Maximal Marginal Relevance (MMR)** [17] diversified search results by taking an existing ranking of documents and rearranging them so that documents dissimilar from previously displayed documents (using a similarity metric) were chosen. Chen and Karger generalised the *PRP* so that it could take into account document dependency, with the intention of both diversifying results and reducing the number that needed to be displayed [61], building on the idea that the novelty of documents is as important as their diversity [36]. In relation to the work in this thesis, Radlinski et al. [23] developed a multi-armed bandit based algorithm for dynamically learning diverse rankings of documents through click observations. Zuccon et al. [62] developed a technique taken from operations research that aimed to find the optimal ranking of k documents so that relevance and diversity were balanced. This formulation is considered a generalization of the portfolio theory of IR, which is covered in more detail in the next section.

2.1.4 Economics in IR

Economic theories lend themselves well to IR problems, as they both involve the interpretation of user/customer behaviour, the balance of risk and reward and the application of statistics and machine learning. For instance, production theory from micro-economics has been applied to interactive IR [63], where the number of is-

sued queries and documents read by a user in a search session are considered the raw materials (the input), the technology used is the search engine, and the output is the cumulative gain of satisfying the user's information need. This model allowed the researcher to find the optimal balance between queries and displayed documents for different types of information search. Another example is the use of game theory, which has been used to model the dynamics between the user and the search system in session search [64] and also to set up a study for search system evaluation [65].

The concept of evaluating for retrieval utility rather than relevance was proposed by Cooper in 1973 [39], where the aim was to maximise a utility function that balanced the costs and benefits of an IR system's actions. An example of utility use in IR was in its combination with hedonic regression to estimate the demand of products online based on the sentiment analysis of item reviews [66]. Azzopardi [67] extended the work on production theory by constructing a utility for the costs of user actions such as examining snippets and reading documents, and used it to understand the trade-offs users make when using a search system. The use of utility functions in IR frameworks to measure user benefit are the subject of Chapter 3.

2.1.4.1 Portfolio Theory

Portfolio theory was devised in 1957 as a means to optimally choose a portfolio of investments based on their risk profiles [68]. This was achieved by investing in assets that were both profitable and negatively correlated with one another, so that a loss of value in one of the assets was balanced by an increase in value in another. The setting of a risk parameter allowed the portfolio holder to maximise their expected returns for a given level of acceptable risk over the whole portfolio.

The theory has been applied to IR, where instead of diversifying a portfolio of assets, a ranking of documents is diversified instead [40]. For instance, if each document has a probabilistic relevance value $r_i = P(R = 1|d,i)$ for a document ranked at position i (where $1 \leq i \leq M$), then the overall expected relevancy of a

ranking is given by

$$\mathrm{E}[\vec{\mathbf{R}}] = \sum_{i=1}^{M} b_i r_i \tag{2.4}$$

where b_i is a weight value representing the rank bias at position i and $\vec{\mathbf{R}} = \langle R_1, \ldots, R_M \rangle$ is a vector of the relevance of each document in the overall ranking. The objective is to maximise Eq. (2.4). The variance of the ranking, and therefore the risk, can be calculated as

$$\mathrm{Var}[\vec{\mathbf{R}}] = \sum_{i=1}^{M} \sum_{j=1}^{M} b_i b_j \sigma_{i,j} \tag{2.5}$$

$$= \sum_{i=1}^{M} b_i^2 \sigma_{i,i} \sigma_{i,i} + 2 \sum_{i=1}^{M} \sum_{j=i+1}^{M} b_i b_j \sigma_{i,j} \tag{2.6}$$

$$= \sum_{i=1}^{M} b_i^2 \sigma_i^2 + 2 \sum_{i=1}^{M} \sum_{j=i+1}^{M} b_i b_j \sigma_i \sigma_j \rho_{i,j} \tag{2.7}$$

where $\sigma_{i,j}$ is the covariance between the documents ranked at positions i and j, σ_i the standard deviation of the document ranked at position i and $\rho_{i,k}$ the *correlation coefficient*. Using this formulation, it can be shown that a suboptimal but efficient way of maximising the value $\mathrm{E}[\vec{\mathbf{R}}]$ while minimising the risk $\mathrm{Var}[\vec{\mathbf{R}}]$ in a ranked list is to first choose the highest ranking document, then for each subsequent rank j, to choose the document that maximises

$$r_j - \lambda b_j \sigma_j^2 - 2\lambda \sum_{i=1}^{j-1} b_i \sigma_i \sigma_j \rho_{i,j} \tag{2.8}$$

where λ is a tunable exploration parameter.

The use of portfolio theory in IR is not restricted only to the case of diversifying search results, but has also been used to trade off the risk of incorrectly performing query expansion [69]. Another application of risk in IR is in the area of learning to rank, where consideration can be given to the robustness of learned models so that they work for tail queries, balancing the risk of generating bad search rankings for tail queries against suboptimal rankings for all queries [70].

2.2 Dynamics

For over 200 years, dynamics has played an important role in the field of engineering, from James Clerk Maxwell's 'Centrifugal Governor' [71] to the stabiliser at the back of the Wright brothers' plane [72]. Physical systems that maintain an equilibrium, such as those described, are designed as dynamic agents that use feedback to keep control in a closed loop. The study of systems like these is known as **control theory** [10] and are modeled using diagrams similar to that in Fig. (1.1).

Control theory largely applies to mechanical systems, but in recent decades has also found use within electronics and software, for instance, to model electronic closed loop flip-flop storage state circuits [73] or in the streaming of audio and video files [74]. Early research in artificial intelligence and robotics used control theoretic principles in the design of models such as the Markov decision process, used for path-finding algorithms [75] or for stable and predictable controllers for robotic parts [76].

This work paved the way for the most recent application of dynamics in computer science, **Reinforcement Learning (RL)**. RL algorithms describe a dynamic process that learns from its past actions and adapts to observations of its environment following those actions over time. Over the years many models for RL have been developed, including the partially observable Markov decision process [77] (covered in Section 2.2.3.2), active learning [78] and the multi-armed bandit [79] (covered in Section 2.2.3.4).

2.2.1 Dynamics in IR

Dynamics also increasingly plays an important role in the design of information retrieval systems. An IR system can be broken down into broadly five elements, the *users*, their *information needs* and the *queries* that they use to represent them, the *documents* being retrieved and their underlying *relevance* to the queries. Each of these elements can be considered as a dynamic agent depending on the IR problem being addressed:

Users A single user can be considered as a dynamic agent in the case where their context or descriptive features may change over time. For instance, during a

search session the user's contextual search preference can be updated based on their observed actions [80] or their physical location [81], or their search history can be used to personalize search results and identify when they are behaving atypically [82]. A population of users may also be considered dynamic, with each different user a new signal in the system. This is the case with online learning to rank and is explored further in Chapter 5.

Information Needs An information need may dynamically evolve over the course of an exploratory search session [83], which is identified as a corollary of the findings in Chapter 6. A user may also have several information needs that are satisfied over their interactions with the search system [84]. Search results diversification is one method for addressing a dynamic or complex information need and there has been some research into learning diverse rankings of documents over time based on user clicks [23], a topic explored further in Chapter 5.

Queries For a fixed information need in session search, a user will issue multiple queries, such as in Table (1.1). Understanding the dynamics of queries in session search is the subject of Chapter 6. The queries that users use for the same information need may change over time depending on the evolution of the language used to describe the need, or the definition of the need itself [85]. By observing these trends in search logs, query suggestion agents can be built that can also take into account user attributes such as gender or age [86, 87].

Documents The content of individual documents can be considered dynamic. For instance, the content of some webpages (such as Wikipedia[1] articles) has been found to change over time [88, 89]. In addition to this, the language used in a document may have changed since the document's creation, with certain terms having a different interpretation nowadays to that originally intended [90]. Collections of documents may also dynamically change over time, for instance, adaptive filtering is used when retrieval occurs over a

[1] `http://wikipedia.org/`

stream of documents [91], and the available documents in a corpus decreases during interactive retrieval when feedback is gained on already displayed documents. This is also the case with the multi-page search scenario explored in Chapters 3 and 4.

Relevance Dynamically changing relevance reflects the dynamics of the world. For instance, a news event such as a natural disaster or the death of a notable person may render documents that were previously relevant as irrelevant and result in different or new documents becoming relevant [92, 93]. Further to this, natural language is dynamic; new words, phrases, colloquialisms, brand names and products change the meanings of queries and terms, resulting in a change to the relevance of documents. For example, users searching for the query `twitter` a decade ago may have been interested in birdsong, whereas now they are more likely to be entering a navigational query to access the popular microblogging website[2] [94]. The changing of relevance over time is a key aspect of the *DIR* framework discussed in this thesis in Chapters 3, 4 and 5.

The challenge of incorporating these dynamic elements into IR demand that search systems upgrade themselves from performing simple retrieval tasks to becoming decision engines that can pick the best choice for information seeking dynamic tasks. Search systems are evolving into natural language, conversational question answering services that need to be able to update relevance based on previous interactions and context [95]. As the world wide web ages the content of web pages changes, queries change meaning, information needs become redefined and user characteristics change.

2.2.2 Relevance Feedback

In order for an interactive or dynamic system to be adaptive, it must acquire relevance signals from the user so that it can update its internal model of the user's preferences. These signals can be directly collected from the user in the form of

[2] `http://twitter.com/`

relevance feedback, which falls broadly into three categories:

- In **explicit** relevance feedback, the user is directly solicited for information regarding the relevance of information items, or else their information need preferences. This is usually manifested through an interactive UI that lets users input a binary label or numerical score for items representing their relevance, usefulness or perhaps irrelevance. Alternative methods of collecting feedback include asking users to answer questions regarding their interests for adaptive filtering [96] or allowing users to select item attributes for filtering in faceted e-commerce search [97].

  What these methods have in common is that the user is interrupted during the search process so that the search system may receive feedback. Despite the proven benefits of incorporating explicit relevance labels into the search process found by Salton [98] and Rocchio [16], a wealth of literature has demonstrated the detrimental effect to users that occurs during the collection of the labels. One study found that inexperienced search users preferred a system using implicit relevance feedback over an explicit one, as it reduced the burden of providing feedback [99]. Another found that in UIs incorporating relevance feedback and re-ranking strategies, users favoured those that gave them the greatest amount of control, disliking systems that automatically re-ranked results or acted unpredictably [100]. An analysis of search log data found that typically less than 10% of users made use of an available relevance feedback option during searches, even when results were often over 60% better as a result [101]. This reluctance of the user in the submitting of direct input into relevance feedback systems has motivated the development of collecting implicit signals instead.

- In **implicit** relevance feedback, signals observed from natural user interactions with the search system are unobtrusively recorded and interpreted as signals of relevance [102]. Research has found a correlation between relevance and a range of user behaviours, including document reading time [103], webpage dwell time in search [104] and scrolling behaviour [105]. A study by

Fox et al. found that the right combination of implicit measures could work as effectively as explicit feedback, although many of the signals in their study (such as the session-based features) were less effective [37]. They also found that document clickthroughs proved one of the strongest implicit signals for relevance feedback, a topic already covered in Section 2.1.2.

- **Pseudo** relevance feedback systems simulate explicit relevance feedback by assuming that the top k ranked documents in a search ranking are relevant. They then perform conventional relevance feedback to improve search results [106, 107]. Pseudo-relevance feedback has been an active area of research in the last twenty years, usually resulting in improved search performance for single ad hoc queries [108, 109].

Once feedback from a user has been observed, an interactive search system can then improve the search experience. Early research in relevance feedback concerned systems that were personalized to each user and dealt with the difficulty for a user to come up with an appropriate query for a given information need. In these systems, after an initial search, a user could explicitly select documents relevant to their information need, resulting in an automatic reformulation of their query into a more relevant one [110]. Modern static search systems also employ interactivity. For instance, query auto-completion [111] systems suggest new queries based on partially typed queries and can also take into account context and past interactions [87]. Query suggestion can also be considered interactive, whereby the search system actively attempts to direct the search session the user is engaged in based on their current query [112].

2.2.2.1 Rocchio Algorithm

The **Rocchio** algorithm is a well-established and important relevance feedback mechanism for improving search results in interactive systems [16, 113]. It follows a two-stage approach utilizing explicit relevance labels collected from the user. In the first stage, the user enters a query and reviews the set of search results returned by the system, labelling which documents they found relevant, and

which they didn't. By representing the query and documents using the vector space model, the search system uses the Rocchio formula to modify the query vector so that it moves closer to the relevant documents and further away from the non-relevant documents. The Rocchio formula is given by:

$$\vec{\mathbf{q}}^* = \alpha\vec{\mathbf{q}} + \frac{\beta}{|D_\mathrm{r}|}\sum_{\vec{\mathbf{d}}\in D_\mathrm{r}}\vec{\mathbf{d}} - \frac{\gamma}{|D_\mathrm{nr}|}\sum_{\vec{\mathbf{d}}\in D_\mathrm{nr}}\vec{\mathbf{d}} \tag{2.9}$$

where $\vec{\mathbf{q}}$ and $\vec{\mathbf{q}}^*$ are respectively the original query vector and its modified version under Rocchio. D_r and D_nr are the set of relevant and non-relevant documents (as labelled by the user) and $\vec{\mathbf{d}}$ is the document vector. α, β and γ weight the effect of the original query, relevant documents and non-relevant documents on query modification. In the second stage, ranking and retrieval occurs using the modified query which usually leads to improved recall. This is shown explicitly in Fig. (3.1b) and is similar to the interactive solution to the multi-page scenario in Fig. (1.5).

2.2.3 Reinforcement Learning

Reinforcement learning algorithms are designed to maximise some cumulative **reward** by making actions in their environment and learning the outcome through a feedback signal. Unlike supervised learning, RL algorithms can only make an observation and thus an inference on the correctness of their action after it has been made, and it is often the case that observations cannot be made on those actions not chosen at each time step. An RL algorithm is a dynamic agent and one that is investigated throughout this thesis as a means for designing algorithms in *DIR*.

2.2.3.1 Markov Decision Processes

A **Markov Decision Process (MDP)** is a stochastic decision process with the Markov Property. An MDP is composed of **agents**, **states**, **actions**, a **reward** function and a **transition probability distribution** [41]. An agent takes inputs from the environment and outputs actions; the actions in turn influence the other states of the environment according to the transition probability distribution. MDPs assign immediate rewards for taking an action at a state. Formally, an MDP is a tuple [114]

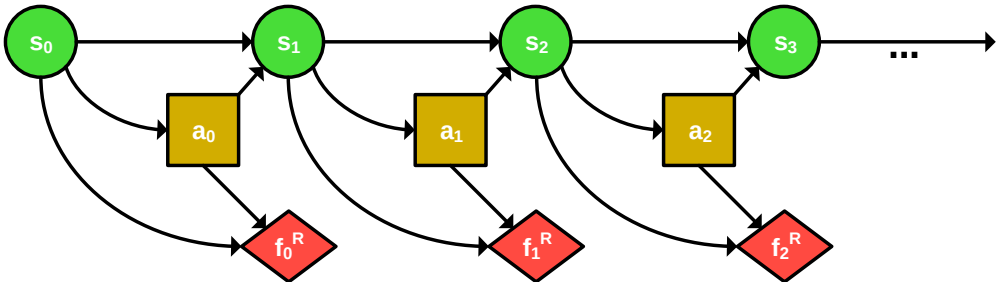


Figure 2.1: The Markov Decision Process influence diagram.

$(\mathscr{S}, \mathscr{A}, f^R, \pi)$, made up of:

States The state $s \in \mathscr{S}$ describes the status or environment that the agent is in at a given moment in time.

Actions The action $a \in \mathscr{A}$ describes the possible changes that the agent can make at a given moment of time given the current state.

Reward The reward function $f^R(s,a)$ is the reward of taking action a when in state s.

Transition Probability The transition probability $\pi(s,a,s')$ denotes the probability of transition from state s to state s' triggered by an action a.

In addition, $\omega \in [0,1]$ can be set as a discount factor for future rewards. The goal of an MDP is to find an optimal policy, i.e. the best sequence of actions $a_0, a_1, \ldots$ that maximises $\sum_{t=0}^{\infty} \omega^t f^R(s_t, a_t)$, the long-term accumulated reward which sums up all (discounted) rewards from the beginning to the end of the process. A typical MDP structure is shown in Fig. (2.1).

2.2.3.2 Partially Observable Markov Decision Processes

A **Partially Observable Markov Decision Process (POMDP)** is a variant MDP that takes into account that the agent may not know which state it is currently in. Generally speaking, it models an agent's belief about its current state based on observations of its environment and the actions available to it. After taking an action and receiving a reward, the agent makes a new observation and updates its belief about its state. The POMDP formulation allows the agent to optimise which action to take depending on its belief state. The POMDP theoretical model can be

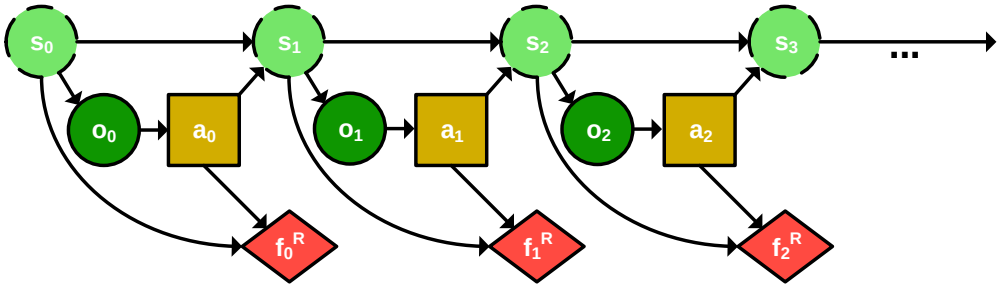


Figure 2.2: The Partially Observable Markov Decision Process influence diagram, where states are no longer observable but their observations are.

represented by the tuple $(\mathscr{S}, \mathscr{A}, f^R, \pi, \mathscr{O}, \Theta, r, \tau)$. The states, actions, rewards and transition probability are defined as for an MDP, the new elements are:

Observations In the POMDP framework, agents cannot determine which state they are currently in directly, but they can make an observation $o \in \mathscr{O}$ which gives some indication of the state they are in.

Observation Function The observation function $\Theta(s,a,o) = P(o|s,a)$ determines the probability of making a particular observation given the hidden state of the agent and the action taken.

Belief The belief is the probability that the agent is in a given state and is given by the function $r(s) = P(s)$.

Belief Update Function The agent maintains a belief about its hidden state given by r. After making an action and receiving an observation, the belief can be updated using the belief update function $r'(s) = \tau\big(r(s),a,o\big)$ which can be estimated by $\sum_{o\in\mathscr{O}} P\big(r'(s)|r(s),a,o\big)P\big(o|a,r(s)\big)$.

The goal of a POMDP is to find an optimal policy of actions that maximises the overall expected reward $\sum_{t=0}^{\infty} \omega^t \sum_{s\in\mathscr{S}} r(s) f^R(s_t,a_t)$. A typical POMDP structure is shown in Fig. (2.2).

2.2.3.3 Dynamic Programming

An optimal solution to an MDP can be found using **dynamic programming**, which is the process of breaking down a difficult problem into sub-problems that can be

solved independently, then combining their solutions to form the overall solution to the main problem. For an MDP, this means choosing the optimal action to take at a particular time step in order to maximise the immediate reward. The Markov property ensures that rewards are not dependent on previous rewards, and so an optimal policy across many time steps can be found.

There are two well-established dynamic programming algorithms for MDPs: *value iteration* is the case where the expected overall state of the dynamic system is set and then optimization occurs backwards until an optimal sequence of decisions is made from the starting point; likewise, in *policy iteration* the optimization occurs from the starting point until the model's reward converges [41]. In addition, there are two well-known variants to this solution: Q-Learning [115] which shows how to build an optimal policy function that does not require knowledge of the underlying model; and Temporal Difference learning [116], where Monte Carlo sampling is combined with an MDP framework to predict the next state and update the model parameters.

In a POMDP the state is unknown, and observations of the state may be misleading or not convey enough information. The solution to a POMDP is made by mapping it onto a continuous state Markov decision process, where the continuous state is the belief state. Through such a transformation, the value function of a POMDP can be expressed in a recursive form similar to an MDP using the **Bellman equation** [77]:

$$V\big(r(s)\big) = \max_{a} \left[f^{R}\left(r(s), a\right) + \omega \sum_{o \in \mathscr{O}} P\left(o' | a, r(s)\right) V\left(r'\right) \right] \tag{2.10}$$

Here, potential observations of the state are marginalised over and r' is the belief state in the next iteration given by the belief update function $\tau\big(r(s), a, o\big)$. The belief update is the most important aspect of the POMDP as it allows one to estimate the state transitions occurring whilst not being able to directly observe them.

2.2.3.4 Multi-Armed Bandits

The **Multi-Armed Bandit (MAB)** [79] comes from a classic statistical resource allocation problem usually described using the analogy of a casino with multiple one-armed bandit slot machines. At each time step, a single machine (or *arm* in the literature) is chosen and a reward drawn from its hidden probability distribution. Meanwhile, the rewards for the other arms remain unknown. Each of the slot machines has a different probability distribution of rewards, and the objective is to find the optimal strategy for playing them in order to maximise the overall accumulated reward over some time horizon.

An analytical, Markovian solution was posited by Gittens in 1979 [43], defining for each one-armed bandit a scalar value known as the **Gittens index**. This index value represented the expected reward for playing an arm until a termination step, and so for each time step, the arm with the highest Gittens index value was the arm chosen to play. The computational intractability of the calculation of the index has lead to the development of tractable solutions that can guarantee asymptotic regret bounds [117].

Regret is the commonly used evaluative framework for multi-armed bandit algorithms, defined as the difference between the sum of rewards of the optimal sequence of arms versus those chosen by the MAB algorithm. Here, the regret ρ is given by

$$\rho = T\left(\max_i r_i\right) - \sum_{t=1}^{T} \hat{r}_t \tag{2.11}$$

where r_i is the reward of the i-th arm and $\hat{r}_t$ is the reward of the arm chosen at time t. In dynamic programming and other optimal control formulations, the objective is to maximise the value function over time, whereas here the goal is to minimise the regret, which represents the loss in potential value caused by choosing incorrectly.

The multi-armed bandit formulation can be likened to the Markov-based models described so far. Like a POMDP, one can consider the reward distribution of each arm as its hidden state, and so the belief state is the estimated expected value

of the reward for that arm. The action in this case is which arm is chosen at each time step, and the reward (and observation) is what is observed from the chosen arm. A key distinction lies in the fact that the hidden distribution of each bandit does not change over time, that is, there is no transition probability function as the arms are independent and fixed. This simplification, as well as the adaptability of MAB algorithms to different scenario settings, makes MABs a versatile tool that are useful in *DIR*, in particular online learning to rank, where only partial information (clicks) can be observed at each iteration.

2.2.3.5 Exploration vs. Exploitation

A key characteristic of the MAB formulation is the balance between exploration and exploitation. In the context of multi-armed bandits, *exploration* refers to the investigation of the hidden distribution of each arm by strategically playing them to observe their reward. Conversely, once this distribution has been reliably determined then *exploitation* occurs, whereby the arm with the estimated highest reward is played.

The estimated reward is typically given by the maximum likelihood estimate

$$\bar{r}_i = \frac{1}{M_i} \sum_{m=1}^{M_i} r_{i,m} \tag{2.12}$$

which is the average reward for arm i played M_i times ($M_i \leq T$). A naïve, purely exploitative algorithm would simply pick the arm with the highest $\bar{r}_i$ at each time step, but could never recover if it picked sub-optimally. The ε-greedy strategy addresses this by introducing a probability ε that at each time step a random arm is chosen, otherwise the best arm so far is chosen with probability $1-\varepsilon$. This solution has been demonstrated to be effective in simple scenarios although it is sensitive to the tuning of the parameter ε and the reward distributions of the arms [118].

A popular and effective alternative is the **Upper Confidence Bound (UCB)** index-based approach [119, 118]. Here, at time t, the arm with the highest index

value

$$\bar{r}_i + \sqrt{\frac{2 \ln t}{M_i}} \tag{2.13}$$

is played. This is the upper Chernoff-Hoeffding confidence interval bound for the estimated reward for probability $1 - \frac{1}{t}$ and the algorithm comes with a regret bound of $\frac{8M}{\Delta*} \ln t + 5M$, where $\Delta*$ is the difference in reward between the optimal arm and the next best arm. As t increases, the upper confidence bound increases logarithmically for those arms not being played. Playing an arm (increasing M_i) causes a linear decrease in this bound. This extra term allows for the exploration of arms throughout the lifetime of the algorithm. Arms with a high average reward are more likely to be played and exploited, resulting in a lower upper confidence bound, allowing for inferior arms to occasionally be played.

2.2.3.6 Multi-Armed Bandit Variations

The multi-armed bandit literature is diverse and the methods described so far are simply the well-known algorithms that work under simple, well-understood problem settings. Different settings have been proposed that are more reflective of practical situations. For example, the multi-play setting has been widely researched for use in news recommendation [120] and signal allocation [121]. In these cases, k arms are chosen and played at each time step rather than a single arm. Other variations of the classic formulation include arms that are dependent on one another [122], arms with non-stationary reward distributions [123], arms that have some probability of not being active at each time step [124] or that have a limited lifespan before being removed from the set of arms [125].

An alternative approach that is important to online learning to rank is the *contextual bandit*. Here, each arm has a feature vector associated with it which is related to its reward distribution. Thus, a contextual MAB algorithm learns both the reward distributions for each arm and also a relevance model for the features. This allows for an improved learning rate, as the model can be used to estimate the reward for uncertain or unplayed (or new) arms, playing those anticipated to be

effective based on their features. This has found application in news article and ad recommendation [126, 120].

Relevant to pairwise learning to rank is the *duelling bandit*. Here, absolute rewards from individual arms are no longer observable. Instead, pairs of arms are chosen at each time step, and the observed 'reward' is the outcome of their comparison, i.e. which has the higher reward [127]. This formulation is useful in cases where there is no natural way of directly measuring rewards but comparisons can be measured. This is applicable to the online learning to rank scenario where a user's subjective perception of the relevance of documents is difficult to collect and interpret, but through interleaved experiments comparisons between ranking systems can be reliably made [128].

Chapter 3

Dynamic Information Retrieval

Theoretical frameworks like the *PRP* and its more recent *Interactive Information Retrieval* variant have guided the development of ranking and retrieval algorithms for decades, yet they are not capable of modeling problems in *Dynamic Information Retrieval* which exhibit the following three properties; an observable user signal, retrieval over multiple stages and an overall search intent. In this chapter a new theoretical framework for retrieval in these scenarios is proposed. Here, a general dynamic utility function for optimizing over these types of tasks is derived, that takes into account the utility of each stage and the probability of observing user feedback. This framework is applied in experiments over TREC data in the dynamic multi-page search scenario as a practical demonstration of its effectiveness and to frame the discussion of its use, its limitations and to compare it against the existing IR frameworks.

3.1 Introduction

The theoretical frameworks that underpin research in IR are based on abstract utility models of user benefit. For instance, the loss function defined in the classic *PRP* [5] leads to justification for the simplest and most prevalent ranking rule in IR; ranking documents in decreasing order of their probability of relevance. A recent counterpart is the Probability Ranking Principle for Interactive IR (*IIR-PRP*) [129], which relaxes the independence assumption in the *PRP*'s model to take into account non-linear decision making. For example, document dependence is a key element in IR

diversification that is not handled by the *PRP* [40].

DIR is defined as exhibiting three characteristics: user feedback, temporal dependency and an overall goal. In this chapter *DIR* is presented as a natural progression in IR research complexity; where early research concerned *static* problems such as ad hoc retrieval, which gave way to *interactive* tasks such as those incorporating relevance feedback [16], finally leading to *dynamic* systems where tasks such as ranking for session search are optimised [130].

From this progression a generalised framework that models the expected benefit to a user of completing a *DIR* task is mathematically formulated. This benefit is represented as a recursive utility function that is goal oriented and adaptive over time. The components of this utility function represent the three *DIR* features: the likelihood of user feedback, a probability of relevance model conditioned on this feedback and an individual stage utility function. The optimization of this recursive utility leads to an optimal policy of actions dependent on user interactions in the dynamic setting.

This utility is shown to be a form of Bellman equation [42] and the framework an instantiation of a POMDP [131]. The components of the utility can also be linked to the cost-benefit parameters of the *IIR-PRP* and the discount-gain functions found in session-based metrics such as *sDCG* [132]. The structure of the utility function and its links to these areas of research give interesting insights into the behaviour of the function in *DIR* problems. These insights, such as how the quality and diversification of rankings vary over multiple stages, are supported by experiments performed using a specific application of the *DIR* utility function over TREC data. The experimental setting is the multi-page search scenario of choosing optimal rankings to display over several search pages for a fixed query [133, 134], a simplified *DIR* problem. As well as being a demonstration of the implementation of each of the components that make up the utility, practical aspects such as the computational complexity are also explored.

3.2 Comparison of IR Frameworks

Before setting up the framework for dynamic information retrieval, *DIR* is considered in the context of existing static and interactive theoretical IR frameworks in order to mathematically identify those features that distinguish it.

3.2.1 Static IR Framework

Definition: A static IR framework is one which models single user interactions, or else multiple *independent* interactions of different search intents. A typical application would be an ad hoc ranking and retrieval system.

The objective of a static system is to choose an **action** a (or sequence of actions $\vec{\mathbf{a}} = \langle a_1, a_2, \ldots \rangle$), each of which has an associated **probability of relevance** r (or $\vec{\mathbf{r}}$ for a sequence of actions) that maximises some **static utility function** $U_S(a,r)$. The action represents a choice that can be made by the system and belongs to some action space $\mathscr{A}$. For example, a may be a query suggestion to display to a user, or $\vec{\mathbf{a}}$ the ranking order of a set of documents for retrieval. The utility function gives value to the action based on its probability of relevance by modeling the benefit of the action to the user. Utilities such as expected **Discounted Cumulative Gain (DCG)** and MAP [135] are examples from document retrieval, rewarding the ranking of relevant documents at high ranking positions.

3.2.1.1 Probability Ranking Principle

A well-established static framework for ranking documents is the **Probability Ranking Principle (*PRP*)**. In 1979, Robertson [5] stated that the effectiveness of a retrieval system is maximised when displaying documents in decreasing order of their estimated probability of relevance: it is most beneficial to display the document that has the highest probability of relevance first, followed by the second highest and so forth. This intuitive result formalised the optimal strategy for displaying ranked documents to users and underlies most modern IR models. Nonetheless, this principle also makes the assumption that one is able to accurately estimate the probability of relevance and also that document relevancies are independent of one another, which is not always the case.

The principle is derived from a utility function for the potential loss when retrieving a document, defined as

$$\text{Loss}(\text{retrieved}|\text{non-relevant}) = \alpha_1 \tag{3.1}$$

$$\text{Loss}(\text{not retrieved}|\text{relevant}) = \alpha_2 \tag{3.2}$$

for parameters α_1 and α_2. Thus, the expected loss if a document d at rank position i with relevance R has been retrieved is

$$\big(1 - P(R|d,i)\big) \times \alpha_1 \tag{3.3}$$

and if the document was not retrieved, then the expected loss is

$$P(R|d,i) \times \alpha_2 \tag{3.4}$$

Thus, the decision of whether to retrieve document d at rank i is determined by whether

$$P(R|d,i)\alpha_2 > \big(1 - P(R|d,i)\big) \times \alpha_1 \tag{3.5}$$

$$\implies P(R|d,i) > \frac{\alpha_1}{\alpha_2 + \alpha_1} \tag{3.6}$$

As a result, documents whose probability of relevance $P(R|d,i)$ falls above this threshold utility value should be ranked in decreasing order of said probability, otherwise they should not be ranked as the expected loss becomes positive.

3.2.1.2 Diversification and the *PRP*

The *PRP* defines $U_S(a,r)$ as a loss minimizing function across pairs of documents (Eq. (3.1) and Eq. (3.2)), which is optimised when ranking documents in decreasing order of probability of relevance (under document independence assumptions). Nonetheless, in instances where result diversity is important, it can be shown that the *PRP* is no longer optimal [40]. This is illustrated in the example in Fig. (3.1a). Here, a simplified vector space model is used to represent ranking and retrieval us-

ing a graph over two term axes. In this case, the query is `apple`, an ambiguous term that can describe three subtopic search intents. Those documents within the ranking $\vec{\mathbf{a}}_{\text{PRP}}$ for the query are retrieved (analogous to ranking under the *PRP*), and as can be seen, in this case only two subtopic preferences are captured. Over a population of users, those seeking information on the `apple logo` subtopic would be dissatisfied.

This is captured probabilistically by supposing that there are two classes of users, $\texttt{user}_1$ and $\texttt{user}_2$, where $\texttt{user}_1$ has twice as many members as $\texttt{user}_2$. Users in the $\texttt{user}_1$ class are satisfied with the `apple logo` and `apple computer` subtopics, but not `apple fruit`, while those in the $\texttt{user}_2$ class are only satisfied with the `apple fruit` subtopic. The action space here is the set of subtopics which is denoted $\mathscr{A} = \{a_1 = \texttt{apple logo}, a_2 = \texttt{apple computer}, a_3 = \texttt{apple fruit}\}$ and the goal is to choose the best ranking of subtopics.

By setting $R_{a_k} = 1$ if a_k is relevant, and $r_{a_k} = P(R_{a_k} = 1)$, then $r_{a_1} = \frac{2}{3}, r_{a_2} = \frac{2}{3}$ and $r_{a_3} = \frac{1}{3}$. According to the *PRP*, the subtopics should be ranked in decreasing order of the probability of relevance, giving the ranking sequence $\vec{\mathbf{a}}_{\text{PRP}} = \langle a_1, a_2, a_3 \rangle$. However, intuitively this is not optimal because users belonging to $\texttt{user}_2$ have to reject two subtopics before reaching their preference [136]. This can be explained mathematically by studying the optimization of the diversity-encouraging metric *Expected Search Length*. One can also derive the same conclusion analogously using the equivalent **Expected Reciprocal Rank (ERR)** or k-call at n measures [61]. In this scenario, $U_S(\vec{\mathbf{a}}, \vec{\mathbf{r}}) = \text{E}[L]_{\vec{\mathbf{a}}}$ which is the summation of all possible search lengths L weighted by their respective probabilities, given as

$$\text{E}[L]_{\vec{\mathbf{a}}} = \sum_i \big((i-1)P(R_1 = 0, \ldots, R_{i-1} = 0, R_i = 1)\big) \tag{3.7}$$

where R_i is the relevance of the subtopic at rank position i. When assuming subtopics are independent, i.e. $P(R_1 = 0, \ldots, R_i = 1) = P(R_1 = 0) \ldots P(R_{i-1} =$

$0)P(R_i = 1)$ the expected search length for ranking $\vec{\mathbf{a}}_{\text{PRP}}$ is

$$\begin{aligned} \mathrm{E}[L]_{\vec{\mathbf{a}}_{\text{PRP}}} &= 0 \cdot r_{a_1} + 1 \cdot r_{a_2}(1 - r_{a_1}) + 2 \cdot r_{a_3}(1 - r_{a_2})(1 - r_{a_1}) && (3.8) \\ &= 0 \cdot (2/3) + 1 \cdot (2/3)(1/3) + 2 \cdot (1/3)(1/3)(1/3) = \mathbf{8/27} && (3.9) \end{aligned}$$

and for a diversified ranking $\vec{\mathbf{a}}_{\text{DIV}} = \langle a_1, a_3, a_2 \rangle$ the expected search length is

$$\mathrm{E}[L]_{\vec{\mathbf{a}}_{\text{DIV}}} = 0 \cdot (2/3) + 1 \cdot (1/3)(1/3) + 2 \cdot (2/3)(2/3)(1/3) = \mathbf{11/27} \quad (3.10)$$

Thus, in this case the *PRP* ranked documents have a shorter expected search path than the diversified ranking. Here, the *PRP* does lead to the optimal ranking under the independence assumption, but when it is removed this is no longer the case. To see this, expected search length for $\vec{\mathbf{a}}_{\text{PRP}}$ and $\vec{\mathbf{a}}_{\text{DIV}}$ is recalculated but this time without the independence assumption:

$$\mathrm{E}[L]_{\vec{\mathbf{a}}_{\text{PRP}}} = 0 \cdot r_{a_1} + 1 \cdot P(R_{a_2} = 1, R_{a_1} = 0) + 2 \cdot P(R_{a_3} = 1, R_{a_2} = 0, R_{a_1} = 0) \quad (3.11)$$

$$= 0 \cdot (2/3) + 1 \cdot 0 + 2 \cdot (1/3) = \mathbf{2/3} \quad (3.12)$$

$$\mathrm{E}[L]_{\vec{\mathbf{a}}_{\text{DIV}}} = 0 \cdot (2/3) + 1 \cdot (1/3) + 2 \cdot 0 = \mathbf{1/3} \quad (3.13)$$

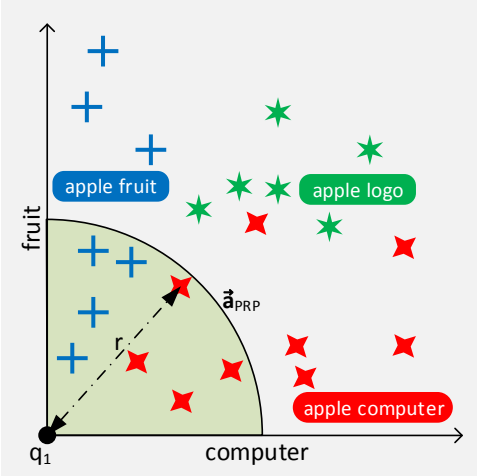
Now it is found that the diversified ranking $\vec{\mathbf{a}}_{\text{DIV}}$ has the shorter expected search length and is thus the optimal ranking, despite the lower probability of relevance for a_3.

3.2.2 Interactive IR Framework

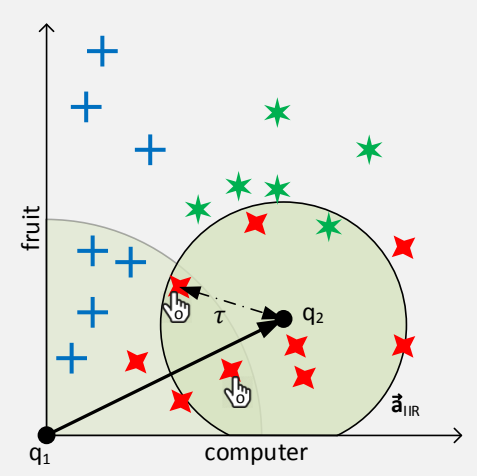
Definition: An interactive IR framework extends a static framework to cover multiple stages of IR. It is *responsive* to feedback from a previous stage but does not anticipate future feedback.

A **stage** represents an interaction with the search system that is distinct from other interactions but belongs to the same search task, for example a sequence of impressions in session search. Generally, an IR system will operate over $1 \leq t \leq T$

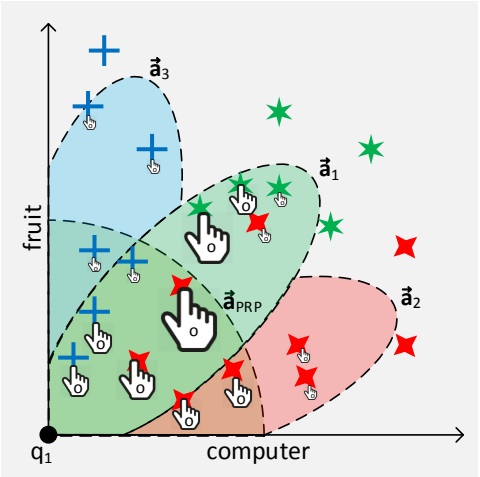
Figure 3.1: An example illustration of document ranking and relevance feedback using the vector space model for query $q_1 =$ `apple`. Documents are given as points over two term frequency axes, `computer` and `fruit`, and can belong to one of three subtopics `apple fruit`, `apple logo` and `apple computer`. The distance between q and each document is inversely proportional to its relevance r. The documents ranked for q_1 or its reformulation q_2 are contained in each circular shape $\vec{\mathbf{a}}$, the area of which could be thought of as the static utility $U_S(\vec{\mathbf{a}}, \vec{\mathbf{r}})$, or U_D the combined area of actions across stages 1 and 2.



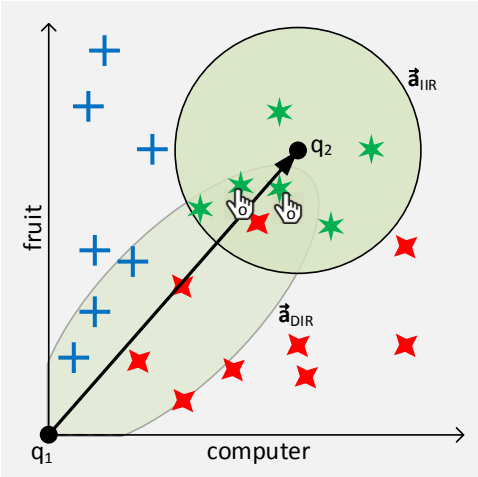
(a) Static IR: Documents within the ranking $\vec{\mathbf{a}}_{\text{PRP}}$ are shown to the user for query q_1, but do not cover all subtopics. Optimally ranking using the *PRP* results in choosing those documents with the highest relevance.



(b) Interactive IR: After relevance feedback is observed (the two click observations o) on the static ranking in Fig. (3.1a), q_1 is modified to q_2. Document relevance for q_2 is now defined by τ and the new interactive ranking given by documents in $\vec{\mathbf{a}}_{\text{IIR}}$.



(c) Dynamic IR: Four potential rankings $\vec{\mathbf{a}}_1$, $\vec{\mathbf{a}}_2$, $\vec{\mathbf{a}}_3$ and $\vec{\mathbf{a}}_{\text{PRP}}$ and their observation probabilities (shown as click observations with likelihood relative to size) for q_1 are explored to find the optimal ranking action for both stages.



(d) Optimal Solution: Action $\vec{\mathbf{a}}_1$ is chosen as the optimal stage 1 ranking $\vec{\mathbf{a}}_{\text{DIR}}$ as it diversely contains documents from all subtopics. As a result, the ranking $\vec{\mathbf{a}}_{\text{IIR}}$ for q_2 is more accurately modified after observing interactive feedback.

stages with T being potentially infinite.

Further to this, an interactive IR framework incorporates user feedback. Feedback is an **observation** signal o (or a sequence of observations $\vec{\mathbf{o}}$) in the space $\mathscr{O}$, that is measurable by the search system. These signals may be *explicit* declarations of the relevance of search items (such as a movie rating), or *implicit* interpretations of user actions (such as document clicks).

The final element of this framework is the **relevance update function** τ where $r_{t+1} = \tau(a_t, r_t, o_t)$. Thus, the objective function for interactive IR at stage $t+1$ can now be represented as

$$\underset{a_{t+1} \in \mathscr{A}}{\operatorname{argmax}} U_S\big(a_{t+1}, \tau(a_t, r_t, o_t)\big) \tag{3.14}$$

The relevance update function τ introduces **temporal dependency** into the framework, without it the objective simply devolves to optimization over the static utility $U_S(a, r)$. This is also the case when finding the optimal first stage action a_1 i.e. when there are not yet any observations. In interactive IR, the optimal action is chosen at each stage as a reaction to the feedback observed in the previous stage and there is no consideration for future utility.

With these features in mind, the vector space example can be extended to the interactive scenario in Fig. (3.1b) by introducing the Rocchio relevance feedback algorithm for interactively re-ranking documents. Here, clicked documents in the *PRP* ranked first stage are used as implicit signals of relevance to modify the user's original query q_1 to q_2. Document re-retrieval occurs using q_2, returning documents using updated relevance scores given by τ, which is a function of q_2 and thus the original ranking $\vec{\mathbf{a}}_{\text{PRP}}$, document relevancies r and observations o. Nonetheless, even in this interactive framework, a user interested in the `apple logo` subtopic would be dissatisfied with both the $\vec{\mathbf{a}}_{\text{PRP}}$ and $\vec{\mathbf{a}}_{\text{IIR}}$ rankings due to a lack of documents for the relevant subtopic.

3.2.2.1 Probability Ranking Principle for Interactive IR

The static *PRP* framework has been extended to an interactive version known as the **Probability Ranking Principle for Interactive Information Retrieval (*IIR-PRP*)** [129]. Here, the assumption of document independence is removed by the definition of a utility function that explicitly incorporates dependence. This utility models a user that makes *choices* while browsing a ranked list of search results. For each choice the benefit and cost to the user is quantified, as well as the probability of the user accepting the choice and also the probability that the choice is relevant. For instance, in ad hoc ranking the cost may be the effort required to read a document snippet and the benefit would be the knowledge gained.

This utility can be described as the expected benefit U of document d_i (document d ranked at position i) given as

$$\mathrm{E}\big[U(d_i)\big] = \omega(i) + P(R_i|d,i)\big(b_i\alpha_i + (1-b_i)\beta_i\big) \tag{3.15}$$

where $\omega(i)$ is the effort required to reach rank i, α_i is the utility gain when the document is relevant and β_i is the utility loss when the document is not relevant. The bias value b_i is the probability that the document at rank i is the correct choice i.e. that $R_i = 1$. When determining the expected benefit of a ranked list of documents, the overall utility is

$$\mathrm{E}\big[U\big(\langle d_1,\ldots,d_M\rangle\big)\big] = \sum_{i=1}^{M}\left(\prod_{j=1}^{i-1}\big(1-P(R_i|d,j)\big)\right)\big(\omega(i)+P(R_i|d,i)(b_i\alpha_i+(1-b_i)\beta_i)\big) \tag{3.16}$$

Fuhr [129] shows that when optimizing over Eq. (3.16), the optimal ranking policy is to rank documents in decreasing order of the utility score

$$\rho(d_i) = \big(b_i\alpha_i + (1-b_i)\beta_i\big) + \frac{\omega(i)}{P(R_i|d,i)} \tag{3.17}$$

3.2.2.2 Interactive Information Retrieval

For clarification, the area of research traditionally known as **Interactive Information Retrieval (IIR)** has an alternative definition to the interactive IR *framework* discussed in this chapter, despite the similarity in name. IIR research explores the complex sequence of interactions a user may have with a search ranking within the static framework [137], largely motivated by the contradictory results found from conventional Cranfield style evaluation [138] and observational user studies [139]. For the remainder of this thesis any reference to interactive IR instead reflects the interactive framework defined in this chapter.

3.3 Dynamic IR Theory

A dynamic system is one which is goal-directed and adaptive to its environment. From this definition one can specify three elements that determine whether an IR system is a dynamic one:

Feedback An observation signal from the user.

Temporal dependency Operation across multiple stages where each stage depends on the previous stage.

Overall goal An objective across all stages.

3.3.1 Dynamic IR Framework

Definition: A dynamic IR framework extends an interactive framework by being responsive to user feedback *and* optimizing for it in advance.

Systems in the interactive IR framework were previously defined as exhibiting both feedback and temporal dependency features, but they are only capable of locally optimizing for a single stage at a time. In contrast, the optimization of a dynamic system will find the optimal sequence of actions for all future interactions. A result of this is that the utility of an individual stage may be reduced so that gains can be made in the utility at a future stage.

Unlike the interactive IR framework, the observation o is unknown when evaluating the utility of future stages in the dynamic IR framework. Instead, the *expected*

utility can be found by marginalizing the utility function over the space of observations $\mathscr{O}$. When doing this the **observation likelihood function** $P(o|a,r)$ must be specified. This gives the expected utility

$$\mathrm{E}\big[U_S(a,r)\big] = \sum_{o \in \mathscr{O}} P(o|a,r) U_S(a,r) \tag{3.18}$$

The observation likelihood function is represented visually in Fig. (3.1c). In dynamic IR, the expected utility of potential first stage rankings (given here as $\vec{\mathbf{a}}_1$, $\vec{\mathbf{a}}_2$, $\vec{\mathbf{a}}_3$ and the static ranking $\vec{\mathbf{a}}_{\text{PRP}}$) are calculated by estimating which documents are likely to receive clicks and the effect this has on the utility of future stages. The *PRP* ranking is simply one among many rank actions that can be considered.

The final component of the *DIR* framework is the **path-discount** function $\omega(t)$. When optimizing over a potentially infinite number of future stages, this helps ensure that a solution exists and also gives greater weight to earlier stage utilities.

By bringing together all of the components described so far, the **utility function for dynamic information retrieval** can be defined as

$$U_D(r_t,t) = \max_{a_t \in \mathscr{A}} \left[U_S(a_t,r_t) + \omega(t) \sum_{o \in \mathscr{O}} P(o|a_t,r_t) U_D\big(\tau(a_t,r_t,o), t+1\big) \right] \tag{3.19}$$

where $U_D(r_T,T) = \max_{a_T \in \mathscr{A}} \big[U_S(a_T,r_T)\big]$ is the static optimization of the final stage. Thus, the objective is to find, through backwards induction, the optimal sequence of actions $\vec{\mathbf{a}}^* = \langle a_1, \ldots, a_T \rangle$ that maximises the **dynamic utility** U_D given in Eq. (3.19). To derive this utility, the dynamic utility has simply been recursively applied to the expected utility from Eq. (3.18).

Through the maximization of the dynamic utility, in the example in Fig. (3.1d) the optimal action for stage 1 is found, which is to diversify the initial ranking so that it retrieves documents belonging to all three subtopics. While this may harm the immediate retrieval utility score, overall the system improves because it can more accurately re-rank results over the subtopic preferences for all users in the next stage.

Table 3.1: Elements of the *DIR* framework.

Element	Description	Examples
a	Action	Query suggestion, ranking of documents
r	Relevance	Query or document relevance
t	Stage	Impression, rank position, page
o	Observation	Click, e-commerce transaction
τ	Relevance Update Function	Rocchio, multi-variate Gaussian
$P(o\|a,r)$	Probability of Observation Function	Click model, eye-tracking distribution
U_S	Static Utility	DCG, ERR
ω	Path-Discount Function	Geometric, path-based

The eight elements of the *DIR* framework: a, r, U_S, t, o, τ, $P(o|a,r)$ and $\omega(t)$, also shown in Table (3.1), are also the elements that define a POMDP, and the dynamic utility function is its corresponding Bellman equation. Intuitively this makes sense, like a POMDP the dynamic IR framework finds an optimal Markovian sequence of actions to maximise a reward (here the static utility) subject to discounting (with ω). The state of the system (the underlying document relevance) is unknown but a belief state (the probability of relevance) is updated according to observations. The key difference from a POMDP is that for dynamic IR there is no defined transition probability between states due to the assumption that the hidden relevance of each document does not change throughout the search task.

3.3.2 Framework Analysis

So far the general framework for dynamic IR has been described but not the setting of its parameters. Here, each component is analysed within the context of dynamic information retrieval.

Relevance As with any framework in information retrieval, the overall aim is to retrieve relevant information items and present them to the user. The intrinsic 'relevance' of an information item is an unknown quality and the subject of most of the research in IR. In the *DIR* framework any document relevance scoring method can be used. For instance, in the application in the next section, five well established relevance scoring techniques are used.

The relevance update function τ is more difficult to define as it depends specifically on the action and observation space of the *DIR* task, for instance in Fig. (3.1b) it depends on the distance of the documents from q_2, which itself depends on q_1, $\vec{\mathbf{a}}_{\text{PRP}}$ and its clicked documents. This dependence allows τ to adapt to the hidden relevance preferences of the user over the course of the search process.

It may not always be clear how to update the relevance score based on a given observation, the most straightforward setting for τ can simply be to set $r_a = 0$ for actions already chosen by the IR system. Because τ enforces the temporal dependency, it is the most important aspect in the dynamic utility because without it the utility is static.

Actions The action space is what distinguishes search tasks from one another and it is the size of this space that dictates the complexity of optimizing over the dynamic utility function. For example, the action space in query suggestion or document ranking is potentially infinite whereas the space of available advertisements in an ad selection problem may be small and finite. In the previous example, the action space is any potential grouping of the documents in the 2D term space (four such groupings are shown in Fig. (3.1c)). Along with τ, the setting of the static utility U_S is important for determining the desirable features of the optimal action sequence $\vec{\mathbf{a}}^*$, such as results diversification.

Observations The observation space is dependent on the action space, its elements representing the user's response to system actions. Each observation must contain some signal of relevance or search intent, otherwise $\tau(a,r,o) = \tau(a,r)$ and there would be no temporal dependency. In some cases the value of the observation likelihood is simply $P(o|a,r) = r$, for instance in search tasks where accurate explicit relevance feedback is guaranteed. Otherwise, in most situations the observations will be click-related and thus the observation probability is the probability of click, as is the case in the example in Fig. (3.1).

Stages Typically, the stages in a *DIR* task represent distinct interactions occurring in a linear time order. In these cases $\omega(t)$ may take a value between 0 and 1 or be set to a monotonically decreasing function that favorably weights the utility scores

of immediate stages. Setting $\omega(1) = 1$ and $\omega(t) = 0$ for $t > 1$ gives the static and interactive scenarios.

Alternatively, a non-linear sequence of interactions (or *search path*) can be modeled as Yang and Lad did with their session-based utility function [140]. For instance in session search, a search path represents a particular sequence of documents examined by the user and the query reformulations made. For the *DIR* framework, the stage t may instead represent a specific search path, and so $\omega(t)$ could be interpreted as the likelihood of this path rather than an explicit discount, penalizing improbable search paths and rewarding likely ones.

The time horizon T dictates the number of advance stages to optimise for. A large time horizon will lead to explorative action strategies that benefit later stages. In the experiments in this chapter, T is set to 2 so that only exploitative optimizations for the immediate next stage are considered.

Dynamic Utility Through the recursive evaluation of the utility function, one can not only learn the optimal sequence of actions to make in the dynamic system, but also learn the optimal action for each possible observation at each stage. If one were to store these in a lookup table ahead of deployment, then the dynamic system would be immediately responsive to user feedback and able to cater to a population of users. Nonetheless, the construction and storage complexity of such a table may prove intractable. Also, the static utility U_S may be set as the dynamic utility function U_D of a nested subproblem in the search task. For example, the utility of choosing an optimal *ranking of documents* may be embedded in the utility for determining an optimal *sequence of rankings* for a user in a session, which itself may be defined within the context of modeling a user's *topic preference* from search sessions in their search history.

3.3.3 Links to Existing Work

Building on the analysis of the *DIR* framework, links between its components and other related work in interactive retrieval and session search can be identified.

3.3.3.1 IIR-PRP

As covered in Section 3.2.2.1, the *IIR-PRP* is a framework designed for interactive IR in the traditional sense. The objective function in *IIR-PRP* balances the costs and benefits of choosing actions within a sequence and bears some similarities to the dynamic utility function. Nonetheless, by lacking any form of user feedback or temporal dependency, in this thesis the model is not described as interactive or dynamic, instead, within the terminology used in this chapter, this means that it is actually a static method.

By using the *DIR* framework notation, the *IIR-PRP* objective function in Eq. (3.16) can be rewritten as

$$U_{\text{IIR}}(\vec{\mathbf{a}},\vec{\mathbf{r}}) = \sum_{i=1}^{M}\left[\prod_{j=1}^{i-1}(1-r_j)\right]\left(\omega(i) + r_i \sum_{o\in\vec{\mathbf{o}}} P(o|a_i,r_i)U_S(a_i,r_i)\right) \tag{3.20}$$

evaluated over a sequence of M actions (usually a ranking of documents). Eq. (3.20) is a generalization of the formula originally defined, where it is recognized that the cost and benefit parameters α_i and β_i are simply utility values, that the probability of whether a user continues searching or not is the observation likelihood, and that the cost of reaching a specific action is the path discount.

Further to this, the *IIR-PRP*'s ranking rule ρ in Eq. (3.17) can similarly be defined in this setting as

$$\rho(\vec{\mathbf{a}}_{1\ldots i},\vec{\mathbf{r}}_{1\ldots i},\omega) = U_S(\vec{\mathbf{a}}_{1\ldots i},\vec{\mathbf{r}}_{1\ldots i}) - \frac{\omega}{r_{a_i}}\prod_{j=1}^{i-1}(1-r_{a_j}) \tag{3.21}$$

where the utility of adding an action a_i to an existing sequence is countered by the path discount and the probability of not finding previous actions relevant. This is implemented in the algorithm for *IIR-PRP* in Algorithm 2 later in this chapter.

3.3.3.2 Session-Based Utility

There have been recent advances in the modeling of user benefit across queries in search sessions. This is in recognition of the fact that ad hoc retrieval often occurs over multiple queries in a session. A simple approach has been to extend discount-

gain metrics such as DCG and average precision, typically associated with static retrieval, across multiple stages. Using the terminology in this chapter, a discount-gain function for a single stage has the form $\sum_{i=1}^{M} \omega(i) U_S(a_i, r_i)$. For example, in the DCG metric the setting would be $\omega(i) = \frac{1}{\log_2(i+1)}$ and $U_S(a_i, r_i) = 2^{r_{a_i}} - 1$. For the **session-DCG (sDCG)** [141] metric, a single layer of recursion is introduced, where

$$sDCG = \sum_{t=1}^{T} \omega(t) U(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) \tag{3.22}$$

and the discount and gain functions are set as $\omega(t) = \frac{1}{\log_{2t}(t+1)}$ and

$$U(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = DCG(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \sum_{i=1}^{M} \underbrace{\frac{1}{\log_2(i+1)}}_{\omega(i)} \times \underbrace{(2^{r_{a_{ti}}} - 1)}_{U_S(a_{ti}, r_{ti})} \tag{3.23}$$

Here, the stages operate across a linear sequence of search rankings. In the **session-Average Precision (sAP)** metric, the path of interaction taken by the user is unknown and so the metric function marginalises over the space of all such paths to find the expected sAP [132].

3.4 Application of *DIR*

Thus far, a theoretical framework for dynamic IR has been formulated and the dynamic utility function U_D given in Eq. (3.19) derived. In this section, the framework is applied to the multi-page search problem in *DIR*. In doing so, the functional settings for the elements in Table (3.1) and their implementation in a workable algorithm can be demonstrated, which gives useful insight into the practical limitations of optimizing over U_D. This algorithm is compared against *PRP* and *IIR-PRP* based approaches in experiments using TREC data, as well as static and interactive variants of the *DIR* objective function. Through this, an understanding of the effect that dynamic utility optimization has on the quality and diversity of rankings in multi-page search can be gained.

3.4.1 Multi-Page Search Problem

The multi-page search scenario is described in more detail in Section 1.2 and demonstrated in Fig. (1.3), Fig. (1.5) and Fig. (1.7). It concerns the ranking of documents over multiple pages of search results [133, 134]. MPS typically models exploratory search queries which are more likely to lead to multi-query sessions and multi-page searches [6] (with one study finding that 27% of such searches occur over multiple pages [142]). In this scenario documents are retrieved for a single query, ranked and then segregated into pages of M documents. On each page, a user may examine and click on documents. An underlying assumption is that the user will return to the results page and move onto the next page, and so a threshold of T pages can be defined over which the user will search. The goal in MPS is to create rankings of relevant documents across T pages. For the pages following the first, document clicks can be used to personalize search rankings, a situation analogous to the example in Fig. (3.1).

In this scenario, each page of search represents a stage in the framework, with T the threshold number of pages. Nominally, T is set to 2 although a larger number of pages is feasible and is explored further in the next chapter. The action sequence $\vec{\mathbf{a}}_t = \langle a_{t1}, \ldots, a_{tM} \rangle$ represents the ranking of documents for ranks 1 to M on page t. Before Eq. (3.19) can be fully implemented, each of its functional components must be defined in the context of MPS.

3.4.1.1 Expected DCG

The static utility in multi-page search is a measure of the quality of the ranking of documents on each page. As with ad hoc ranking and retrieval, this can be evaluated using a metric such as DCG, MAP or ERR. In the absence of relevance judgements, one can instead find the *expected* metric value which uses probabilities of relevance instead [135]. In the application in this chapter, the static utility is set as the expected DCG function, given by

$$U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \sum_{i=i}^{M} \frac{2^{r_{a_{ti}}} - 1 + 2^{r_{a_{ti}}-1} \log^2(2) \mathrm{Var}[r_{a_{ti}}]}{\log(i+1)} \tag{3.24}$$

This utility also takes into consideration the variance of the document's probability of relevance.

3.4.1.2 Examination Hypothesis

In multi-page search the observations are document clicks, which are regarded as an implicit signal of the relevance of a document to the user. Thus, the clicks from previous search pages can be utilised to update the probability of relevance model and personalize the document rankings for future pages.

For a ranking of M documents, the observation space $\mathscr{O}$ in MPS for a particular page is the combination of binary click events for each document in the ranking. This is denoted as the observation vector $\vec{\mathbf{o}} = \langle o_1, \ldots, o_M \rangle$ where $o \in \{0,1\}$. This could naïvely be set to the uniform distribution $P(o|a,r) = \frac{1}{|\mathscr{O}|}$ but eye-tracking studies and click model research indicates otherwise. In this application, the simplest model is used, the *Examination Hypothesis* model [38] in Eq. (2.1).

This model supports the eye tracking research by inferring that the probability of a click on a document in a ranked list is equal to the product of its probability of relevance and the bias of its rank position. Thus, the probability of a sequence of clicks is given by

$$P(\vec{\mathbf{o}}|\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \prod_{i=1}^{M} (b_i r_{a_{ti}})^{o_i} (1 - b_i r_{a_{ti}})^{1-o_i} \tag{3.25}$$

where b_i is a rank bias parameter. In the implementation in this chaprer, Eq. (3.25) is set as the observation likelihood function and $b_i = \frac{1}{\log(i+1)}$ which is the discount value used in the expected DCG utility in Eq. (3.24).

3.4.1.3 Conditional Multivariate Gaussian Distribution

Once a sequence of click observations for a ranking of documents has been obtained, the probability of relevance distribution for the remaining documents can be updated. This is achieved by defining the distribution of all the probabilities of relevance for all documents in the collection as a multivariate Gaussian distribution

$$R \sim \mathscr{N}(\vec{\mathbf{r}}, \Sigma) \tag{3.26}$$

where R is their collective random variable, $\vec{\mathbf{r}}$ the vector of mean relevance scores and Σ the covariance matrix over the documents. $\vec{\mathbf{r}}$ may be set as any relevance score and Σ may be set using document similarity or other correlation scores (which is investigated further in the next chapter). If $\vec{\mathbf{r}}$ represents a probability of relevance, then the distribution can be set as a *truncated* multivariate Gaussian bounded between 0 and 1. If it is not possible to define the distribution of a relevance score, then the distribution of the mean of multiple relevance scoring techniques can be derived, resulting in an approximately Gaussian distribution that incorporates multiple signals of relevance. It is this approach that is taken in this chapter, where $\vec{\mathbf{r}}$ and Σ are set as the means and variances of the retrieval scores from five well-known techniques, with the diagonal elements from Σ used as variance values for the utility calculation in Eq. (3.24).

Modeling the relevance distribution in this way allows one to conditionally update the probabilities of relevance $\vec{\mathbf{r}}$ based on click observations. For a given rank action $\vec{\mathbf{a}}_t$ (which includes both clicked and non-clicked documents in the ranking), the remaining non-ranked documents are denoted as $\backslash\vec{\mathbf{a}}_t$ and the distribution parameters are partitioned as

$$\vec{\mathbf{r}} = \begin{bmatrix} \vec{\mathbf{r}}_{\backslash\vec{\mathbf{a}}_t} \\ \vec{\mathbf{r}}_{\vec{\mathbf{a}}} \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{\backslash\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t} & \Sigma_{\backslash\vec{\mathbf{a}}_t\vec{\mathbf{a}}_t} \\ \Sigma_{\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t} & \Sigma_{\vec{\mathbf{a}}\vec{\mathbf{a}}} \end{bmatrix} \tag{3.27}$$

The mean relevance scores and covariance matrix can then be updated for non ranked documents using the formulae

$$\vec{\mathbf{r}}_{\backslash\vec{\mathbf{a}}_t} = \vec{\mathbf{r}}_{\backslash\vec{\mathbf{a}}_t} + \Sigma_{\backslash\vec{\mathbf{a}}_t\vec{\mathbf{a}}_t}\Sigma^{-1}_{\vec{\mathbf{a}}\vec{\mathbf{a}}}(\vec{\mathbf{o}} - \vec{\mathbf{r}}_{\vec{\mathbf{a}}_t}) \tag{3.28}$$

$$\Sigma_{\backslash\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t} = \Sigma_{\backslash\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t} - \Sigma_{\backslash\vec{\mathbf{a}}_t\vec{\mathbf{a}}_t}\Sigma^{-1}_{\vec{\mathbf{a}}_t\vec{\mathbf{a}}_t}\Sigma_{\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t} \tag{3.29}$$

and observations $\vec{\mathbf{o}}$. Thus, for given actions and observations, the functions above can be used to define a new conditional multivariate Gaussian distribution of the probability of relevance of the remaining documents, given as $R_{t+1} \sim \mathcal{N}(\vec{\mathbf{r}}_{\backslash\vec{\mathbf{a}}_t}, \Sigma_{\backslash\vec{\mathbf{a}}_t\backslash\vec{\mathbf{a}}_t}|\vec{\mathbf{a}}_t, \vec{\mathbf{o}})$. For the multi-page search setting, $\tau(a,r,o)$ is defined as the

relevance update function in Eq. (3.28).

3.4.1.4 Geometric Discount

The final component required for the *DIR* application is the discount function $\omega(t)$. In the multi-page scenario, the utility of a linear sequence of document rankings is measured rather than the path-based behaviour of users. As such, the simple discount used in a POMDP is adopted, setting $\omega = \lambda$ (which is effectively setting it as the geometric discount $\omega(t) = \lambda^{t-1}$ due to the recursion of the dynamic utility).

Here, one can consider $\omega(t)$ as the probability of the user visiting page t. When $\lambda = 0$, the assumption is that only the first page will be visited, and when $\lambda = 1$ all pages are equally likely and given equal weight. The optimal setting for λ will vary depending on the type of searches being performed as well as the corpus and quality of results.

3.4.2 DIR-MPS

After defining each of the functional components of the dynamic IR framework for the multi-page search scenario, the **DIR-MPS** algorithm is presented in Algorithm 1. This algorithm is a direct implementation of the recursive utility function U_D in Eq. (3.19) that determines the optimal sequence of document rankings to display for each page. It is a value iteration algorithm that searches the space of potential rankings at each iteration to determine the optimal one, whose value is calculated based on all potential click observations and their subsequent optimal future time steps.

It is worth noting that Algorithm 1 and the described settings for the *DIR* framework elements are one such instantiation of the framework in the multi-page search scenario. The motive in this chapter is not to develop a state of the art new ranking technique but rather to demonstrate the application of the framework to a *DIR* problem. A different instantiation is given in Chapter 4.

3.4.2.1 Dynamic Utility Approximation

The DIR-MPS algorithm features a number of approximation techniques that increase its computational efficiency as a way to counteract the inherent complexity of

Algorithm 1 The DIR-MPS Algorithm

function DIR-MPS($t, \vec{\mathbf{r}}, \mathscr{A}$)
  if $t = T+1$ **then return** $[0, \langle\rangle]$
  end if
  $\vec{\mathbf{a}}^*_t = \langle\rangle; \vec{\mathbf{a}}^*_{t+1} = \langle\rangle$
  loop $i \leftarrow 1$ to M ▷ Sequential Ranking Decision
    $\vec{\mathbf{a}} = \vec{\mathbf{a}}^*_t; u^* = 0$
    for all $a \in \mathscr{A} \backslash \vec{\mathbf{a}}$ **do**
      $\vec{\mathbf{a}}_t = \langle \vec{\mathbf{a}}, a \rangle$
      $u_t = U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t)$ ▷ Eq. (3.24)
      for all $\vec{\mathbf{o}} \in \mathscr{O}$ **do**
        $\vec{\mathbf{r}}_{t+1} = \tau(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}, o)$ ▷ Eq. (3.28)
        $[u_{t+1}, \vec{\mathbf{a}}_{t+1}] = \text{DIR-MPS}(t+1, \vec{\mathbf{r}}_{t+1}, \mathscr{A} \backslash \vec{\mathbf{a}}_t)$
        $u_t = u_t + \lambda \cdot P(\vec{\mathbf{o}} | \vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) \cdot u_{t+1}$ ▷ Eq. (3.25)
      end for
      if $u_t > u^*$ **then**
        $u^* = u_t; \vec{\mathbf{a}}^*_t = \vec{\mathbf{a}}_t; \vec{\mathbf{a}}^*_{t+1} = \vec{\mathbf{a}}_{t+1}$
      end if
    end for
  end loop
  return $[u^*, \langle \vec{\mathbf{a}}^*_t, \vec{\mathbf{a}}^*_{t+1} \rangle]$
end function

the *DIR* framework (discussed further in Section 3.4.3). Firstly, the action space of potential rankings is reduced by employing a greedy **sequential ranking decision** policy. That is, for each page the optimal document to rank at each position is found one by one. For example, this is achieved by first setting $M = 1$, then finding the document a^* that maximises $U_S(a, r_a)$. Then after this document is fixed, M is set to 2 and the next optimal action in the sequence that maximises $U_S(\langle a^*, a \rangle, \vec{\mathbf{r}}_{\langle a^*, a \rangle})$ is found. Continuing in this fashion allows one to find an approximately optimal ranking for a single page, one document at a time, greatly reducing the computational complexity.

A property of the probability distribution given in Eq. (3.25) also means that the observation space can be greatly reduced. This distribution follows Zipf's law, where a few of the click combinations contribute towards most of the probability mass. In fact, from early DIR-MPS evaluation experiments it was found that typically around 15% of the combinations contributed to 95% of the aggregated

Algorithm 2 The IIR-PRP-MPS algorithm

function IIR-PRP-MPS($M, T, \lambda, \mathscr{A}, \vec{\mathbf{r}}$)
  $\vec{\mathbf{a}}^* = \langle\rangle$
  loop $i \leftarrow 1$ to $M \times T$
    $a^* = \text{argmax}_{a \in \mathscr{A} \setminus \vec{\mathbf{a}}^*} \rho\left(\langle \vec{\mathbf{a}}^*, a \rangle, \vec{\mathbf{r}}, \lambda\right)$ ▷ Eq. (3.21)
    $\vec{\mathbf{a}}^* = \langle \vec{\mathbf{a}}^*, a^* \rangle$
  end loop
  return $\vec{\mathbf{a}}^*$
end function

probability. As such, in this chapter's implementation of DIR-MPS the observation space was restricted to only the most probable click combinations that cumulatively sum to 0.95, trading off the potential 5% inaccuracy for speed.

Finally, it can be shown that when ranking over a single stage, the expected DCG utility function is maximised when documents are ranked according to the *PRP* [135]. This is exploited to increase the efficiency of the DIR-MPS algorithm by ranking the threshold page (where there is no longer a future temporal dependency) according to the *PRP* over the conditionally updated probabilities of relevance.

3.4.2.2 IIR-PRP-MPS

In the experiments, DIR-MPS is directly compared against rankings created from the applied *PRP* and *IIR-PRP* ranking rules. With the *Probability Ranking Principle* one can simply rank documents in decreasing order of the probability of relevance across T pages. However, the *IIR-PRP* has no existing direct application to the MPS scenario. Instead, the definition of the ranking function ρ given in Eq. (3.21) is used to create the IIR-PRP-MPS algorithm shown in Algorithm 2.

Here, the sequential ranking rule is also employed to build up an optimal ranking over all pages, one document at a time, by selecting the document that has the highest ρ value for each rank. Thus, there is some dependency on previously ranked documents, which is not possible in the *PRP*, but like the *PRP* there is no way to take into account user feedback or update the probabilities of relevance.

3.4.3 Practical Limitations

The general computational complexity of the optimization of U_D can be shown to be PSPACE-Complete (through its connection to *POMDP*s). For small T and observation and action spaces this can be reasonable, but typically these spaces may be impractically large.

For example, an IR task such as information filtering or music recommendation may operate over potentially infinite time steps. In these cases the discount factor and threshold T are important. Further, the observation space may not be as well defined as that in the multi-page search scenario where $|\mathscr{O}| = 2^M$, for example, the space of possible reformulations for a query or 2D gaze positions in eye-tracking. Finally, the action space can be difficult to optimise over as is the case with DIR-MPS, where the sequential ranking decision reduces the size of the action space from $O(N!/(N-TM)!)$ to $O(TNM-TM^2)$ for a collection of N documents. The application in this chapter serves to demonstrate that such approximations may be needed when working with the *DIR* framework, especially given that the optimization of U_D is not guaranteed to be tractable, and an optimal solution may not exist depending on the particular problem settings.

3.4.4 Experiment

To gain insight into the application of the dynamic IR framework in the multi-page search scenario, and to compare with the other theoretical frameworks, an experiment was conducted using the WT10g, AQUAINT and ClueWeb09 datasets, the details of which are included in Table (3.2). These test collections were chosen as they were designed for evaluating ranking and retrieval algorithms and were easily extended to the multi-page problem. The WT10g dataset allowed for testing the theoretical frameworks in the standard ad hoc ranking and retrieval environment. The Robust data consisted of difficult to rank ad hoc queries which were hypothesized to be more likely to require several pages of search results. The diversity track data allowed for testing of the hypothesis that dynamic optimization leads to increased diversification in ad hoc retrieval. A drawback to using these datasets is that they lacked interaction data, which is not needed when optimizing for probable clicks in

Table 3.2: Overview of the three TREC test collections.

Name	Task	# Docs	Topics
WT10g	TREC 9 Web Track	1,692,096	451-500
AQUAINT	Robust 2005	1,033,461	50 difficult Robust 2004 topics
ClueWeb09	Diversity Task 2009/10	503,903,810	1-100 (461 subtopics)

the *DIR* framework, but important in the interactive setting.

On each collection, the top 100 documents were retrieved for each topic scored using each of the TF-IDF, BM25, Jelinek-Mercer, Dirichlet and Two-Stage language model retrieval methods from the Indri[1] search engine. The documents were pooled and subsequently scored over all the techniques. This gave an average of 193 ranked documents per topic each with 5 relevance score values. After min-max normalization, each score was averaged to give the probability of relevance vector $\vec{\mathbf{r}}$ and covariance matrix Σ. The dependencies in this covariance matrix reflected the level of agreement between the different retrieval methods rather than direct correlations between the documents themselves i.e. similarly ranked documents were positively correlated with one another. Finally, those documents that had the top 30 mean relevance scores were selected. These were then used by the experiment algorithms to create rankings for two pages of search results with ten documents on each.

These documents were ranked according to the baseline *PRP* approach and also the already described IIR-PRP-MPS and DIR-MPS algorithms. An interactive version of DIR-MPS (called IIR-MPS) was also investigated, that ranked the first page of results according to the *PRP* and then optimised a ranking for the second page of results by marginalizing over potential clicks using Eq. (3.18). In this case, only second page optimization occurred based on the feedback from the statically ranked first page. Also, a static version of DIR-MPS (called S-MPS) was investigated, that removed feedback from U_D entirely to give the objective function $U_S(\vec{\mathbf{a}}_1,\vec{\mathbf{r}})+\lambda U_S(\vec{\mathbf{a}}_2,\vec{\mathbf{r}})$. 'Perfect click' variants of the dynamic (DIR-MPS^C) and interactive (IIR-MPS^C) algorithms were also investigated, where the hidden relevance

[1] `http://www.lemurproject.org/indri.php`

labels were interpreted as clicks on the first page of results (i.e. pseudo-relevance), giving the optimal observation setting and an upper bound on performance for the second page.

To evaluate the quality of the rankings, MAP, nDCG and ERR were measured for each page. For the DIR-MPS and IIR-MPS algorithms, an optimal 2nd page ranking for every click combination was generated in the observation space, giving different metrics scores for each. In these cases, the reported page two metric scores are averages over the page two scores for all click combination based rankings. This highlights an open area for research; the definition of evaluation metrics for *DIR* that can take into account all of the potential rankings in a dynamic system. The session-based metrics sDCG (defined in Eq. (3.22)) and sAP were also measured to evaluate performance over both pages, although it is worth noting that these metrics were designed for session search rather than multi-page search. Finally, $\alpha-$nDCG [36], Intent-Aware Precision (IA-Precision) [143] and Intent-Aware ERR (ERR-IA) [144] were measured for scoring the diversity of rankings in the ClueWeb09 collection.

The results of the experiments are shown in Table (3.3). For the Web Track and Robust collections, one can observe that the 1st page losses of the dynamic techniques (when compared to the *PRP* and IIR-PRP-MPS baselines) are made up for by gains in the second page, significantly so on the WT10g dataset. Nonetheless, in these ad hoc ranking scenarios it is clear that the static *PRP* and *IIR-PRP* frameworks are still very effective.

The results are different with the diversity track data. The metric scores for this data in Table (3.3) were calculated using relevance judgements from all subtopics. The opposite relationship between page scores can be seen here, with DIR-MPS having higher scores for the 1st page and losses in the second, except for ERR which is significantly improved across both pages. There is further evidence of this with the diversity metric scores in Table (3.4), where it is clear that diversification is occurring in the first page and less so in the second. This backs up the intuition (in Fig. (3.1d)) that a dynamic technique will initially diversify results to improve

Table 3.3: nDCG, MAP and ERR scores for pages 1 and 2 of the search results. Static, interactive and dynamic algorithms are grouped. The results shown are those for the optimal value of λ in each collection, found by repeating the experiment for values in the range $[0,1]$. The maximum score for each metric on each page is given in boldface. A [1] or [2] indicates that the result is significantly better than the *PRP* or *IIR-PRP-MPS* baseline scores respectively using the Wilcoxon signed-rank test ($p < 0.05$).

		Page 1			Page 2		
Collection	Algorithm	nDCG	MAP	ERR	nDCG	MAP	ERR
Web Track (WT10g) $\lambda = 0.5$	PRP	**0.338**	**0.167**	**0.169**	0.133	0.025	0.053
	IIR-PRP-MPS	0.330	0.162	0.162	0.166	0.041	0.078
	S-MPS	0.295	0.134	0.130	0.226	**0.070**	0.242[12]
	IIR-MPS	**0.338**	**0.167**	**0.169**	0.125	0.025	0.103[1]
	IIR-MPS^C				0.154	0.040	0.151[1]
	DIR-MPS	0.235	0.091	0.092	0.212[1]	0.054[1]	0.289[12]
	DIR-MPS^C				**0.230**[1]	0.059	**0.297**[12]
Robust (AQUAINT) $\lambda = 0.5$	PRP	0.624	**0.107**	**0.398**	0.552	0.061	0.294
	IIR-PRP-MPS	**0.629**	**0.107**	**0.398**	0.514	0.052	0.288
	S-MPS	0.608	0.096	0.388	**0.595**	**0.066**	**0.887**[12]
	IIR-MPS	0.624	**0.107**	**0.398**	0.519	0.050	0.737[12]
	IIR-MPS^C				0.554	0.057	0.690[12]
	DIR-MPS	0.548	0.063	0.304	0.575	0.065	0.656[12]
	DIR-MPS^C				0.553	0.058	0.697[12]
Diversity (ClueWeb09) $\lambda = 0.8$	PRP	0.402[2]	**0.049**[2]	0.199	**0.476**	0.052	0.265
	IIR-PRP-MPS	0.384	0.046	0.193	0.468	0.051	0.257
	S-MPS	0.388	0.041	0.193	0.465	**0.054**	0.358[12]
	IIR-MPS	0.402[2]	**0.049**[2]	0.199	0.431	0.042	0.353[12]
	IIR-MPS^C				0.436	0.042	0.345[12]
	DIR-MPS	**0.451**[2]	0.047	**0.238**[2]	0.445	0.042	**0.373**[12]
	DIR-MPS^C				0.426	0.037	0.356[12]

future rankings, and also helps explain the losses in performance of the 1st page in the other datasets (which do not have subtopic relevance judgements). The reduced diversity of the 2nd page indicates that it is more tightly focused on the user's subtopic preference.

It appears that the diversity task is more suited as an application of the dynamic IR framework. This is evidenced by the optimal settings for λ in each collection. For the ad hoc ranking task in the WT10g and AQUAINT collections, the setting for

Table 3.4: α-DCG, IA-Precision and ERR-IA scores for page 1 and 2 search results from the diversity track data. The maximum score for each metric on each page is given in boldface. A ^{2} indicates that the result is significantly better than the *IIR-PRP-MPS* baseline score using the Wilcoxon signed-rank test ($p < 0.05$).

	Page 1			Page 2		
Algorithm	α-DCG	IA-Prec	ERR-IA	α-DCG	IA-Prec	ERR-IA
PRP	0.360	**0.083**	0.239	**0.420**	0.085	**0.294**
IIR-PRP-MPS	0.345	0.077	0.230	0.404	0.086	0.269
S-MPS	0.352	0.078	0.233	0.417	**0.089**	0.280
IIR-MPS	0.360	**0.083**	0.239	0.377	0.079	0.243
IIR-MPS^C				0.379	0.080	0.236
DIR-MPS	**0.403**^{2}	0.082	**0.270**	0.400	0.079	0.264
DIR-MPS^C				0.386	0.077	0.254

λ gives greater weight to the 1st page of results, rewarding immediately effective rankings. Whereas in the diversity task, the utility of the 2nd page has a larger effect on the overall utility, encouraging diversity.

Further to this, the interactive variant scored highly with session based metrics on the Robust dataset in Table (3.5), but otherwise the static techniques were optimal, even for the diversity task. This may partly be due to the application of a session-based metric to the multi-page scenario and also the inability of the metric to take into account the user interaction. Finally, it can also be seen that the 'perfect click' variants generally outperform their counterparts (except over the diversity data), indicating that the 2nd page ranking can be improved when high quality clicks are observed.

In summary, by its nature the *DIR* approach to multi-page search places greater emphasis on different stages of the search task. It seems that it may not be suited to all search environments i.e. ad hoc search. In such cases the static approaches can be more effective. Nonetheless, the dynamic IR framework has other desirable properties such as the diversification and personalization of results over time.

3.5 Related Work

Throughout this chapter the dynamic IR theory has been presented within the context of the surrounding and aforementioned literature, in this section are those areas

Table 3.5: sAP and sDCG for both pages of results. The results are grouped identically to those in Table (3.3).

		Both Pages	
Collection	Algorithm	sAP	sDCG
Web Track (WT10g) $\lambda = 0.5$	PRP	0.097	1.326
	IIR-PRP-MPS	0.101	1.347
	S-MPS	0.095	1.236
	IIR-MPS	0.097	1.291
	IIR-MPS^C	**0.102**	**1.353**
	DIR-MPS	0.069	1.022
	DIR-MPS^C	0.072	1.027
Robust (AQUAINT) $\lambda = 0.5$	PRP	**0.085**	5.735
	IIR-PRP-MPS	0.083	5.680
	S-MPS	0.083	**5.749**
	IIR-MPS	0.081	5.543
	IIR-MPS^C	0.084	5.729
	DIR-MPS	0.065	4.921
	DIR-MPS^C	0.062	4.909
Diversity (ClueWeb09) $\lambda = 0.8$	PRP	**0.051**[2]	**1.883**[2]
	IIR-PRP-MPS	0.048	1.808
	S-MPS	0.049	1.787
	IIR-MPS	0.047	1.783
	IIR-MPS^C	0.047	1.787
	DIR-MPS	0.046	1.859
	DIR-MPS^C	0.044	1.839

of the related work not already discussed.

For instance, the settings of the components in the *DIR* framework for multi-page search cover a wide area of research in IR. Firstly, the examination hypothesis model used is just one of a number of probabilistic click models that could have been employed, including the click-chain model [57] and even a POMDP-based model [145]. Other path-based discount functions have been explored in the literature [146] as well as other multi-stage utilities and metrics such as Time-Based Gain [147]. Related work on using Markov chains to measure the utility of rankings at each time step is a potential method for evaluating *DIR* problems [148]. Further to this, the identification of the dynamic IR framework as a POMDP raises the pos-

sibility of using established techniques such as the Witness algorithm [149] to find optimal action policies.

The work on defining the elements of a POMDP in session search [130] is a close relation to this work, though focusing more on the testing of particular settings of *DIR* components in the session search scenario rather than explicitly gaining an understanding of the framework and components themselves. Nonetheless, their work contains an evaluation of algorithms that fall under the *DIR* framework including one similar to DIR-MPS.

This work differs from the literature in that: 1) this is the first work to define the characteristics that distinguish dynamic IR from the other theoretical IR frameworks, 2) the dynamic utility is the generalization of many existing ranking utilities and incorporates many elements of IR research such as click models, and 3) the effectiveness of the framework and the static frameworks in different scenarios is confirmed in the experiments.

3.6 Conclusion

In this chapter, a theoretical framework for *Dynamic Information Retrieval* has been established. By contrasting with *static* and *interactive* frameworks, the three characteristics that define dynamic IR systems have been identified; user feedback, temporal dependency and an overall goal. This motivated the derivation of the dynamic utility function U_D, which has its roots in the POMDP formulation. The components of this utility can be directly implemented using elements from existing research which are applied in the DIR-MPS algorithm, an example instantiation designed for the multi-page search problem. The experiments confirm that in this scenario, one of the effects of optimizing for U_D is the diversification of search results. Otherwise, they also demonstrate that for other scenarios the *PRP* and *IIR-PRP* frameworks are still effective.

Chapter 4

Dynamic Multi-Page Search

In this chapter, the *DIR* framework is once again applied to the problem of multi-page search. Here, potentially more accurate and efficient settings for the framework elements are explored: for instance, utility approximation is achieved using Monte Carlo sampling, and a single relevance scoring technique combined with document similarities are used in the relevance model. TREC evaluations demonstrate that the optimal ranking strategy can naturally optimise the trade-off between exploration and exploitation and maximise the overall IR metric score over all pages. Significant gains are made against a number of well-established ranking baselines from the areas of diversification and relevance feedback. Once again, it is shown that the optimal ranking policy is to choose a diverse, exploratory ranking to display on the first page, followed by a personalised re-ranking of the remaining pages based on the user's feedback.

4.1 Introduction

A well-established problem in the ad hoc search scenario in IR is that a user is often unable to precisely articulate their information need in a suitable query. In such cases, users can be found to issue short, exploratory queries that roughly encapsulate their need, followed by a rigorous exploration of the retrieved documents [150]. This exploration can occur across multiple queries in a session (the focus of Chapter 6), otherwise across multiple pages of search. These short queries are often ambiguous such as with the `jaguar` example in Fig. (1.3) or the `apple` example

in Fig. (3.1). Ranking for these types of queries can be made easier when given access to context such as relevance feedback, and improved by introducing diversity.

In this chapter, the *DIR* framework is utilised in an algorithm for dynamic ranking that uses feedback from the top-ranked documents in ad hoc retrieval to contextually re-rank the remaining documents. The scenario considered is the familiar multi-page search scenario, where the search results are split into multiple pages that the user traverses by clicking a *'next page'* button on the SERP. The multivariate Gaussian relevance model is once again used for the relevance update function, but the other components of the *DIR* framework are implemented using other techniques, for instance, document covariance is determined using cosine similarity, a better measure of document similarity than the previous chapter's relevance score distribution, and Monte Carlo sampling is used to reduce the framework complexity. The dynamic programming approach is used to optimise the balance between exploration (maximising the learning of the documents' relevance) and exploitation (choosing only the most relevant documents). The new algorithm is tested on TREC datasets and the results show that the method significantly outperforms other strong baselines across multiple pages.

4.2 Related Work

Shen [151] designed a system similar in scope to this whereby a re-ranking of documents occurred whenever a user clicked on a webpage and then returned to the SERP, rather than explicitly navigating to the next page. Aside from this difference in user interface, no exploration occurred on the original ranked list and as such, document dependence and diversity were not factored into the formulation. This is an example of an interactive approach to this problem. A similar interactive system displayed dynamic drop-down menus containing new document rankings where users could click on individual search results [152], allowing the user to navigate SERPs in a tree-like structure. Again, this method only took into account feedback on one document at a time and did not incorporate diversity or document exploration into its original ranking.

The work most related to this work is the SurfCanyon[1] search engine [153] which utilises user signals such as clickthroughs in order to interactively improve search for a user across multiple pages and queries. This is achieved by using a mixture of different user signals and the method incorporates a number of interactions including user interface changes, results-branching and page re-ranking. The approach in this thesis is a more subtle one, only affecting re-rankings that occur later in the user's returned results with the intention of remaining invisible in order not to disrupt the user's search experience. Furthermore, it uses the state of the art dynamic IR framework presented in this thesis. Since the publication of this work, Luo et al. [130] have re-implemented this algorithm for the related session search problem and compared it to other state of the art techniques, with mixed results.

This work differs from the literature in that: 1) the dynamic re-ranking of documents only occurs whenever the user navigates to the next page in an SERP, making it less disruptive than other interactive techniques, 2) active exploration occurs in the original ranking, leading to diverse results that improve the performance of the system over all pages, and 3) the algorithm is based on the dynamic IR framework.

4.3 Problem Formulation

The formulation in this chapter concerns the exploratory, multi-page dynamic search problem, where a user performs an initial search which returns a set of ranked results that take into account the covariance of the documents, allowing for diversification and exploration in order to learn an optimal ranking for the next pages. The formulation models a user who navigates the SERP in a typical way, clicking on documents or providing explicit relevance ratings. Upon clicking the *'next page'* button to indicate that they would like to view more results, the feedback obtained up until that point is combined with the document covariance to re-rank the search results on the next page.

In traditional IR ranking, documents would be ordered under a static or interactive framework such as the *PRP* or the *IIR-PRP*. Here, the *DIR* framework is

[1] http://www.surfcanyon.com/

used instead. In particular the probability distribution of the document relevance is given by the multi-variate Gaussian distribution from Eq. (3.26), which is equivalent to assuming that the document probability of relevance estimate is correct subject to some Gaussian noise. As a result, the relevance belief update function from Eq. (3.28) (as well as the covariance matrix update function in Eq. (3.29)) are also used in this formulation to update the probabilities of relevance after observing user feedback.

In the previous chapter, $\vec{\mathbf{r}}$ and Σ were calculated by averaging the scores from five different document scoring methods. Here, only a single model is considered. The well-known BM25 score is used for $\vec{\mathbf{r}}$, although other relevance scoring methods could be used such as the vector space model, Dirichlet etc. The covariance matrix Σ is approximated by measuring the cosine and Jaccard document similarity metrics for all document pairs, although topic clustering techniques could also be used [154].

For each query, the search system displays M documents for each results page. The rank action is denoted as the action vector $\vec{\mathbf{a}}$ where a_{ti} is the index of the document retrieved at rank i on the tth page i.e. $\vec{\mathbf{a}}_1$ is the page 1 (initial) ranking and $\vec{\mathbf{a}}_2$ is the page 2 (re-ranked based on feedback) ranking. $\vec{\mathbf{o}}_t = \langle o_{t1}, \ldots, o_{tK} \rangle$ is the vector of feedback observations obtained from the user while interacting with page t, where K is the number of documents given feedback with $0 \leq K \leq M$, and o_{ti} is the relevance feedback for the document ranked at position i.

For each page, the static utility is a simplified version of the expected DCG score from Eq. (3.24), denoted here as

$$U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \lambda_t \sum_{i=1}^{M} \frac{\mathrm{E}[R_{a_{ti}}]}{\log_2\left(i + (t-1)M + 1\right)} \tag{4.1}$$

where $\mathrm{E}[R_{a_{ti}}] = \hat{r}_{ti}$ is the estimated relevance of the document chosen for rank position i on result page t. The overall utility across all pages is thus $\sum_{t=1}^{T} U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t)$. This utility was chosen as it is simple and rewards finding both the most relevant documents and also ranking them in the correct order and it does not require the

variance of the documents, which was not considered in the experiments in this chapter for simplicity. The rank weight $\frac{1}{i+(t-1)M+1}$ is used to give greater weight to ranking the most relevant documents in higher positions and on earlier pages and is similar to the weight used in the sDCG metric.

In this formulation, the path-discount is set using the tunable mixture model parameter $\omega(t) = \lambda_t$ with $\lambda_t \geq 0$, which is used to adjust the importance of results pages and thus the level of exploration in the initial page. When λ_1 is set close to zero, the initial ranking is chosen so as to maximise the next page's ranking utility and thus the priority is primarily on exploration [155]. In the opposite extreme, setting $\lambda_t = 1 \ \forall t$ means that there is no exploration at all and the documents are ranked according to the *PRP* on every page.

From the experiments in the previous chapter as well as the findings from this chapter's experiments, it appears that setting the page threshold T to 2 (only considering the immediate next page) gives the best results. To simplify the derivations in this chapter, T is set as 2 although all of the results and algorithms can be easily generalised to the case where $T > 2$.

4.3.1 User Feedback

Important elements of the *DIR* framework are the observations $\vec{\mathbf{o}}$ and their probability distribution $P(\vec{\mathbf{o}}|\vec{\mathbf{a}},\vec{\mathbf{r}})$. In the previous chapter, clicks and non-clicks were considered as signals of relevance and the examination hypothesis model in Eq. (3.25) was used as their distribution. In this chapter, the interpretation of the observations is given greater flexibility by allowing for direct user ratings to be used instead. In this case, clicks can be interpreted as binary ratings.

In lieu of a specific model for finding the probability of the feedback observations, the Gaussian distribution over the ranked documents' relevance estimates can be used instead,

$$o_a \backsim \mathcal{N}(\hat{r}_a, \hat{\sigma}_a) \tag{4.2}$$

The probability of observing a particular observation vector $\vec{\mathbf{o}}$ depends on both

the probability distribution function of the Gaussian distribution in Eq. (4.2) and also the static utility's DCG rank bias weight $1/\big(log_2(i+1)\big)$, which is derived as:

$$P(\vec{\mathbf{o}}|\vec{\mathbf{a}},\vec{\mathbf{r}}) = \frac{1}{(2\pi)^{K/2}|\Sigma_{1'}|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{\mathbf{o}}-\vec{\mathbf{r}}_{1'})^T\Sigma_{1'}^{-1}(\vec{\mathbf{o}}-\vec{\mathbf{r}}_{1'})\right\} \prod_{i=1}^{M} \frac{\big(\log_2(i+1)-1\big)^{1-\varepsilon_i}}{\log_2(i+1)} \tag{4.3}$$

where $\varepsilon_i = 1$ if the document at rank i has received a user judgement, otherwise 0. If $\varepsilon_i = 1 \ \ \forall i$ up until rank M, then $K = M$. $\vec{\mathbf{r}}_{1'}$ and $\Sigma_{1'}$ refer to the probabilities of relevance vector and covariance matrix of only those documents in rank action $\vec{\mathbf{a}}_1$ that have feedback in $\vec{\mathbf{o}}$.

4.3.2 Dynamic Utility Optimisation

By finding the optimal ranking actions that maximise the static utility in Eq. (4.1) across all pages, the dynamic utility function $U_D(\vec{\mathbf{r}},t)$ can be recursively derived as

$$U_D(\vec{\mathbf{r}}_1,1) = \max_{\langle\vec{\mathbf{a}}_1,\ldots,\vec{\mathbf{a}}_T\rangle}\left[\sum_{t=1}^{T}\lambda_t\sum_{i=1}^{M}\frac{\mathrm{E}[R_{a_{ti}}]}{\log_2\big(i+(t-1)M+1\big)}\right] \tag{4.4}$$

$$= \max_{\vec{\mathbf{a}}_1}\left[\lambda_1\vec{\mathbf{r}}_1\cdot\boldsymbol{W}_1 + \max_{\vec{\mathbf{a}}_2}\mathrm{E}\left[\sum_{t=2}^{T}\lambda_t\vec{\mathbf{r}}_t\cdot\boldsymbol{W}_t\middle|\vec{\mathbf{o}}\right]\right] \tag{4.5}$$

$$= \max_{\vec{\mathbf{a}}_1}\left[\lambda_1\vec{\mathbf{r}}_1\cdot\boldsymbol{W}_1 + \mathrm{E}\left[U_D(\vec{\mathbf{r}}_2,2)\middle|\vec{\mathbf{o}}\right]\right] \tag{4.6}$$

where $\boldsymbol{W}_t = \left\langle \frac{1}{\log_2(1+(t-1)M)},\ldots,\frac{1}{\log_2(M+(t-1)M)}\right\rangle$ is the DCG weight vector. By explicitly setting $T=2$ and λ as a mixture model parameter, Eq. (4.6) is simplified to the dynamic objective function

$$U_D(\vec{\mathbf{r}}_1,1) = \max_{\vec{\mathbf{a}}_1}\left[\lambda\vec{\mathbf{r}}_1\cdot\boldsymbol{W}_1 + \max_{\vec{\mathbf{a}}_2}\left[(1-\lambda)\int_{\vec{\mathbf{o}}}\vec{\mathbf{r}}_2\cdot\boldsymbol{W}_2 P(\vec{\mathbf{o}}|\vec{\mathbf{a}}_1,\vec{\mathbf{r}}_1)\mathrm{d}\vec{\mathbf{o}}\right]\right] \tag{4.7}$$

4.3.3 Monte Carlo Sampling

As already discussed in Section 3.4.3, the dynamic utility can be intractable to solve directly. This is the case with the dynamic utility in Eq. (4.7), where the integral cannot be directly solved for more than three documents. Instead, an approximate solution to the integral can be found by using Monte Carlo sampling, changing the

objective function to

$$U_D(\vec{\mathbf{r}}_1,1) \approx \max_{\vec{\mathbf{a}}_1}\left[\lambda\vec{\mathbf{r}}_1\cdot\boldsymbol{W}_1+\max_{\vec{\mathbf{a}}_2}\left[(1-\lambda)\frac{1}{Z}\sum_{\vec{\mathbf{o}}\in\hat{\mathscr{O}}}\vec{\mathbf{r}}_2\cdot\boldsymbol{W}_2P(\vec{\mathbf{o}}|\vec{\mathbf{a}}_1,\vec{\mathbf{r}}_1)\right]\right] \tag{4.8}$$

where $\hat{\mathscr{O}}\subseteq\mathscr{O}$ is a sampled subspace of the observation space of possible feedback vectors $\vec{\mathbf{o}}$ and $Z=|\hat{\mathscr{O}}|$ is the sample size. The summation in Eq. (4.8) can be easily calculated and Z tuned to trade-off accuracy and efficiency.

4.3.4 Dynamic Exploratory Search

The **Dynamic Exploratory Search (DES)** algorithm is given in Algorithm 3. This algorithm searches through potential page 1 and page 2 rankings to find a sequence that optimises the dynamic utility across both. As with the DIR-MPS algorithm in Algorithm 1, a sequential ranking decision is used to reduce the complexity of finding an optimal ranking for page 1. Likewise, the maximum utility for the second page is obtained when the remaining documents are ranked in decreasing order of $\vec{\mathbf{r}}_{\setminus\vec{\mathbf{a}}}$ i.e. the updated probabilities of relevance for the remaining documents given $\vec{\mathbf{o}}$. Unlike DIR-MPS, this algorithm creates a sample from the observation space using function `CREATESAMPLE` when determining the page 2 utility, and also uses loops rather than recursion due to the fixed number of pages. The setting of K can be assigned a priori or set to M for ease of implementation. If $K<M$, then dynamic exploration only occurs for the first K documents in page 1, then the remainder are ranked according to the *PRP*.

4.4 Experiment

The DES algorithm was evaluated over a series of experiments, testing the setting of its parameters λ, M and T and comparing it against a number of baseline algorithms. The scenario tested in the experiments was a user engaging in multi-page search, this meant generating T pages of rankings of M documents. Unless otherwise stated, M was set to 10 as it provided reasonable and meaningful results within the scope of the test dataset and was reflective of actual search rankings. The DES algorithm was used to generate an optimal ranking of documents for the 1st page

Algorithm 3 The DES (Dynamic Exploratory Search) algorithm

function DES($\vec{\mathbf{r}}, \Sigma, K, Z, \lambda$)
  $\vec{\mathbf{a}} = \text{ARRAY}(M)$
  loop $i \leftarrow 1$ to K ▷ For each rank position
    $U_D = \text{ARRAY}(N)$ ▷ Utility score for each document
    loop $j \leftarrow 1$ to N ▷ For each document
      if $d_j \in \vec{\mathbf{a}}$ **then continue** ▷ Ignore if document already ranked
      end if
      $a_i = d_j$
      $\hat{\mathscr{O}} = \text{CREATESAMPLE}(Z, \vec{\mathbf{r}}, \Sigma, \vec{\mathbf{a}})$ ▷ Create sample space
      sampleAvg $= 0$
      for all $\vec{\mathbf{o}} \in \hat{\mathscr{O}}$ **do** ▷ For each observation in sample
        $\vec{\mathbf{r}}_{\setminus\vec{\mathbf{a}}} = \vec{\mathbf{r}}_{\setminus\vec{\mathbf{a}}} + \Sigma_{\setminus\vec{\mathbf{a}}\vec{\mathbf{a}}}\Sigma_{\vec{\mathbf{a}}}^{-1}(\vec{\mathbf{o}} - \vec{\mathbf{r}}_{\vec{\mathbf{a}}})$ ▷ Eq. (3.28)
        $\vec{\mathbf{a}}_2 = \text{SORT}(\vec{\mathbf{r}}_{\setminus\vec{\mathbf{a}}}, \text{descend})[1 \rightarrow M]$
        sampleAvg $+= \vec{\mathbf{r}}_{\vec{\mathbf{a}}_2} \cdot \boldsymbol{W}_2 \times P(\vec{\mathbf{o}}|\vec{\mathbf{a}}, \vec{\mathbf{r}}_{\vec{\mathbf{a}}})$ ▷ Eq. (4.3)
      end for
      sampleAvg $\setminus= Z$ ▷ Average sample value
      $U_D[i] = \lambda \vec{\mathbf{r}}_{\vec{\mathbf{a}}} \cdot \boldsymbol{W}_1 + (1-\lambda) \times \text{sampleAvg}$ ▷ Eq. (4.8)
    end loop
    $\vec{\mathbf{a}}_i = \text{INDEX}(\text{MAX}(U_D))$ ▷ Sequential Ranking Decision
  end loop
  $\vec{\mathbf{a}}_{K+1 \rightarrow M} = \text{SORT}(\vec{\mathbf{r}}_{\setminus\vec{\mathbf{a}}}, \text{descend})[1 \rightarrow M - K + 1]$
  return $\vec{\mathbf{a}}$
end function

that took into account anticipated relevance feedback. Following this, the relevance judgements of the underlying dataset were used as explicit feedback to update the rankings for the remaining pages. This was similar to the 'perfect click' variants of the DIR-MPS algorithm in the previous chapter.

The DES algorithm was compared with a number of methods representing the different properties of ranking under the *DIR* framework: the baseline static *BM25* ranking under the *PRP*, which was also used as the relevance model for the DES algorithm; an interactive variant of the BM25 ranking that used the conditional model update in Eq. (3.28) for the second page ranking, denoted *BM25-U*; the *Rocchio* algorithm, which used the baseline BM25 ranking for the first page and used relevance feedback to generate a new ranking for the second page; and the Maximal Marginal Relevance (*MMR*) method [17] and its interactive variant *MMR-U*, which diversified the first page using the BM25 ranking and covariance matrix, and ranked

the second page according to the BM25 relevance scores or the conditional model update scores respectively.

The experiments were performed over three TREC collections. The WT10g collection was chosen because it contained graded relevance judgements which could simulate the use of explicit, non-binary relevance feedback for the update function. Similarly, the TREC8 collection was used for its binary relevance judgements that could represent implicit clickthroughs. Finally, the 2004 Robust track was used, which consisted of difficult to rank queries that may result in exploratory search.

For each query (or topic) in the datasets, the 200 documents with the highest BM25 scores were chosen for re-ranking using the DES and other baseline algorithms. Each dataset contained 50 topics, and relevance judgements for those topics were used to evaluate the performance of each algorithm. The judgements were also used to simulate the feedback of users after viewing the first page, allowing for the document relevance in the interactive techniques to be updated for ranking in the second page. The Monte Carlo sample size Z was set to 5000 for all experiments, resulting in a comprehensive sampling of the observation space and an accurate estimate of the second page utility.

Four IR metrics were used to measure performance: precision@K, recall@K, nDCG@K and MRR. These metrics were measured for the first page ($K = M$) and also the first and second pages combined (i.e. $K = 2M$). Precision, recall and nDCG are commonly used IR metrics that are used to compare the effectiveness of different rankings. As this investigation focused on the exploratory behaviour of the DES algorithm, sub-topic related diversity measures were not used in these experiments. However, the MRR was used as a risk-averse measure, where a diverse ranking should typically yield higher scores [40, 61]. Further to this, the session-based metrics used in the previous chapter were not measured.

4.4.1 The Relevance Vector and Covariance Matrix

Unlike the experiments in the previous chapter, here only one relevance score distribution was considered. The BM25 model was used to define the prior relevance

score vector $\vec{\mathbf{r}}$. The normalisation function in Eq. (4.9) was used to scale the BM25 scores onto the range of the relevance judgements found in the dataset

$$\vec{\mathbf{r}} = \frac{\boldsymbol{x} - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})} \cdot \alpha \tag{4.9}$$

where $\boldsymbol{x}$ is the vector of BM25 scores and α is the range of score values i.e. $\alpha = 1$ for binary clickthrough feedback, $\alpha > 1$ for graded explicit feedback ratings.

The covariance matrix Σ represents the uncertainty and correlation associated with the estimations of the document scores. Due to only using one relevance distribution to score documents in this experiment, there was no corresponding covariance matrix readily available. Instead, it was reasonable to assume that the covariance between two relevance scores could be approximated instead using document similarities i.e. two documents with similar content are likely to be relevant to the same queries. Two popular similarity measures were investigated, **Jaccard similarity** and **cosine similarity**. The experiments showed that the latter had a much better performance and so it was used in the remainder of the experiments. The variance of each document's relevance score is set to be a constant in this experiment so as to demonstrate the effect of document dependence on search results, and owing to the difficulty in modeling score variance.

4.4.2 Exploration with λ

In this first experiment, the recall@K, precision@K, nDCG@K and MRR metrics were measured for $K = 10$ and $K = 20$ i.e. metric scores for the first page, and for the first and second page combined. The parameter λ, which indicated the level of exploration that occurred on the first page, was investigated to observe how it affected the performance of the DES algorithm, in particular, the average gains that could be expected over both pages. To calculate the *overall* average gain, the average gain of the DES algorithm over the baseline BM25 ranking

$$\frac{\text{metric@}K(\text{DES}) - \text{metric@}K(\text{BM25 baseline})}{K} \tag{4.10}$$

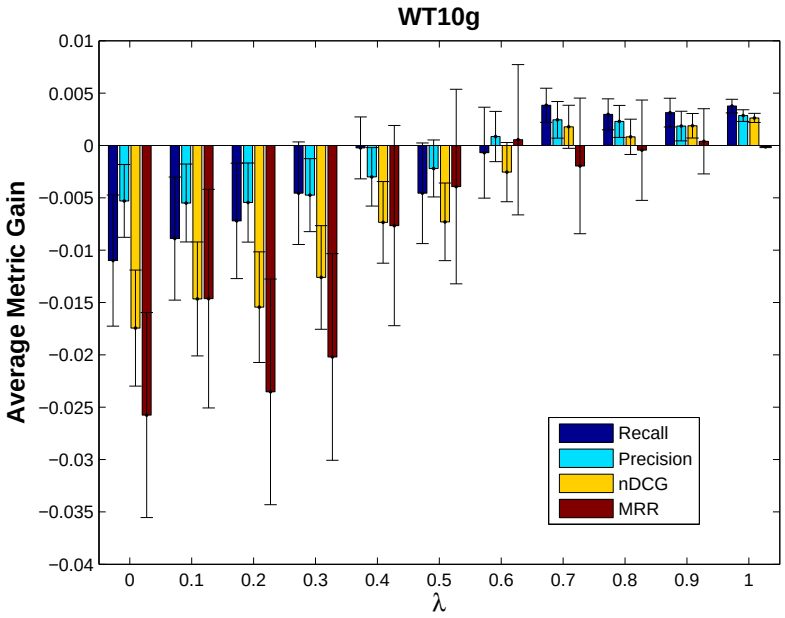


Figure 4.1: The average metric gain of the DES algorithm over the BM25 baseline with 95% confidence intervals. Each bar represents the average gain for each metric for a given value of λ for the WT10g dataset. Positive values indicate gains made over the baseline algorithm whereas negative values indicate losses.

was calculated for $K = 10$ and $K = 20$, and the two values added. This measured the value of gains over both pages compared to just the first page.

In Fig. (4.1), Fig. (4.2) and Fig. (4.3), the variation of the average gain for different values of λ can be seen across the different datasets. The most notable observation is that the value of λ has a definite effect on performance; for smaller values ($\lambda < 0.5$), where the algorithm places more emphasis on displaying an exploratory, diverse ranking on the first page, there is a marked drop in performance across all metrics. As λ increases, performance improves until gains are made. This highlights that too much exploration can lead to detrimental performance in the first page which isn't recovered by the optimised ranking in the second page, indicating the importance of tuning the parameter correctly. On the other hand, the optimal setting for λ is typically less than 1, indicating that some exploration is beneficial and can lead to improved performance across all metrics.

Also, it can be seen that the different datasets display different characteristics with regard to the effect that λ has on performance. For instance, for the difficult to

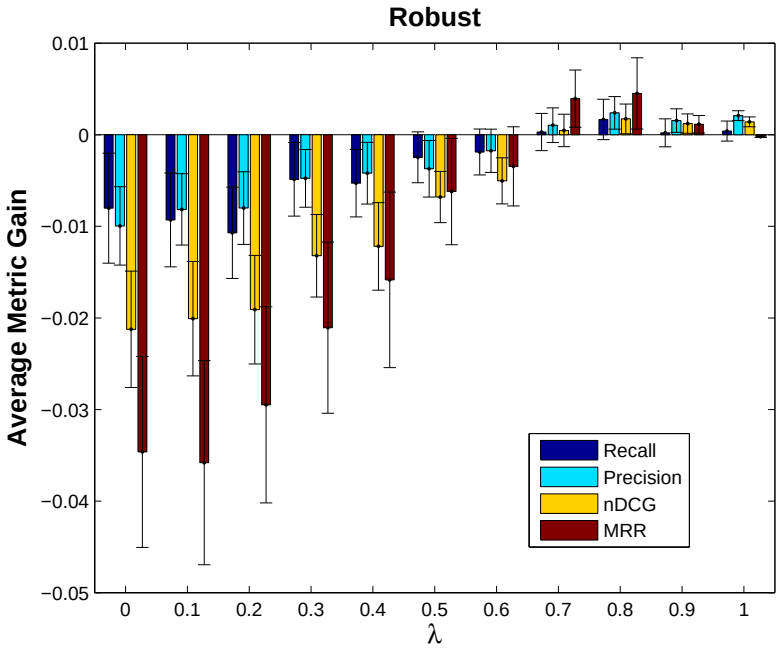


Figure 4.2: Gain results similar to those from Fig. (4.1) for the Robust dataset.

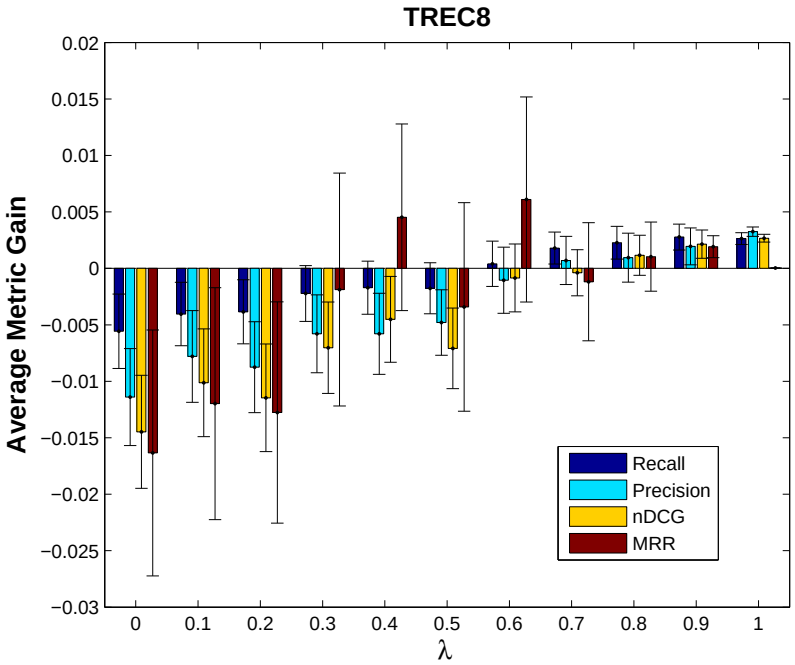


Figure 4.3: Gain results similar to those from Fig. (4.1) for the TREC8 dataset.

rank Robust dataset the optimal setting is $\lambda = 0.8$, indicating that exploration is not very beneficial in this case, possibly due to the lack of relevant documents in the dataset for each topic. Likewise, the setting of $\lambda = 0.9$ is optimal for the TREC8 dataset, although there is greater variation, possibly owing to the easier to rank data. Finally, the WT10g dataset also showed variation in the optimal setting for λ, which could be a result of the explicit, graded feedback available during re-ranking. For this dataset, the optimal setting was chosen as $\lambda = 0.7$. The differing datasets represent different types of query and search behaviour, and illustrate how λ can be tuned to improve performance over different types of search. It is worth noting that this setting of λ is different from that found for the same dataset in Chapter 3 owing to the different definitions of the parameter.

4.4.3 Comparison with Baselines

In the previous experiment the optimal settings for λ were found for each dataset. Likewise, the optimal settings for λ for the MMR algorithm (which is used to balance the influence of document similarity and dissimilarity) were similarly found, giving $\lambda = 0.8$ for the WT10g dataset, $\lambda = 0.8$ for the Robust dataset and $\lambda = 0.9$ for the TREC8 dataset. In this next experiment, the DES algorithm was directly compared with the described baseline algorithms. The experiment was repeated as before over two pages of results with $M = 10$ and the four metrics were measured at $K = 10$ and $K = 20$. It is worth noting that the differences in MRR@10 and MRR@20 were due to those rankings where a relevant document wasn't located in the first M documents. The results can be seen in Table (4.1) and Table (4.2).

From the tables it can be seen that the first page metric scores for DES are generally lower than the other algorithms, which is to be expected when document exploration and diversification is occurring. Nonetheless, the DES algorithm can be seen to outperform the diversifying MMR algorithm, particularly over the risk-averse MRR metric. Conversely, the metric scores for both pages almost always show a significant improvement over the BM25, MMR and Rocchio algorithms. Aside from demonstrating the overall improvement possible against the *static* BM25 and MMR methods, gains are also made against the *interactive* Roc-

Table 4.1: Recall and precision measured at $M = 10$ (first page) and $M = 20$ (first and re-ranked second page) for each algorithm and for each dataset. A superscript number refers to a metric value significantly above the value of the correspondingly numbered baseline in the table (using the Wilcoxon Signed Rank Test with p = 0.05). Boldface metric scores are the highest for that metric across algorithms in that dataset.

			Recall@		Precision@	
			10	20	10	20
WT10g	#	DES	0.229	**0.454**^{13}	0.356	**0.394**^{135}
	1	BM25	**0.233**	0.365	**0.368**	0.321
	2	BM25-U	**0.233**	0.444	**0.368**	0.381
	3	MMR	0.194	0.365	0.326	0.321
	4	MMR-U	0.194	0.410	0.326	0.378
	5	Rocchio	**0.233**	0.400	**0.368**	0.332
Robust	#	DES	**0.231**	**0.390**^{13}	0.385	**0.397**^{123}
	1	BM25	0.228	0.361	**0.392**	0.333
	2	BM25-U	0.228	0.369	**0.392**	0.378
	3	MMR	0.218	0.361	0.377	0.333
	4	MMR-U	0.218	0.379	0.377	0.388
	5	Rocchio	0.228	0.358	**0.392**	0.364
TREC8	#	DES	0.180	**0.377**	0.439	**0.470**
	1	BM25	**0.185**	0.311	**0.455**	0.398
	2	BM25-U	**0.185**	0.364	**0.455**	0.464
	3	MMR	0.181	0.311	0.439	0.398
	4	MMR-U	0.181	0.364	0.439	0.457
	5	Rocchio	**0.185**	0.360	**0.455**	0.404

chio technique over both pages. Non-significant gains are also shown over the BM25-U and MMR-U variants. Considering that both methods used the multivariate Gaussian relevance update function from the *DIR* framework (in particular, the BM25-U algorithm is simply the case where $\lambda = 1$ in the DES algorithm), this indicates that the update function is effective across a number of techniques and able to correctly respond to user feedback and generate a superior ranking on the second page that is tuned to the user's information need.

This experiment was also repeated for $M = 5$ documents on each page. This was to test the hypothesis that the DES algorithm could still rank documents ef-

Table 4.2: nDCG and MRR measurements. This table is structured identically to Table (4.1).

			nDCG@		MRR@	
			10	20	10	20
	#	DES	0.382	**0.476**^{135}	0.590	0.591
WT10g	1	BM25	**0.394**	0.417	**0.603**	**0.605**
	2	BM25-U	**0.394**	0.472	**0.603**	0.603
	3	MMR	0.342	0.417	0.582	**0.605**
	4	MMR-U	0.342	0.441	0.582	0.583
	5	Rocchio	**0.394**	0.439	**0.603**	**0.605**
	#	DES	0.459	**0.561**^{135}	**0.665**	**0.667**
Robust	1	BM25	**0.464**	0.471	0.632	0.638
	2	BM25-U	**0.464**	0.501	0.632	0.634
	3	MMR	0.443	0.471	0.625	0.638
	4	MMR-U	0.443	0.501	0.625	0.626
	5	Rocchio	**0.464**	0.473	0.632	0.638
	#	DES	0.464	**0.523**	**0.653**	**0.658**
TREC8	1	BM25	**0.473**	0.462	0.640	0.645
	2	BM25-U	**0.473**	0.517	0.640	0.646
	3	MMR	0.462	0.462	0.647	0.645
	4	MMR-U	0.462	0.511	0.647	0.652
	5	Rocchio	**0.473**	0.489	0.640	0.645

fectively despite having less documents available from which to gather relevance feedback. Table (4.3) contains a summary of the results for the DES algorithm and the significant gains made over the algorithms from Table (4.1). The results show that the DES algorithm is still effective over reduced page sizes, despite the decrease in available relevance feedback.

4.4.4 Page Threshold

Throughout this chapter, the problem formulation has been simplified by setting the threshold number of pages T to 2. In this experiment, the threshold was extended to $T = 3$. The experiment was repeated as before using $M = 5$ documents on each page; this was so that the running time of the experiment could be decreased, and because the results in Table (4.3) showed similar results when M was set to 5

Table 4.3: Metrics measured at $M = 5$ (first page) and $M = 10$ (both pages) for the DES algorithm on each dataset. The superscript numbers refer to significantly improved values over the baselines, as described in Table (4.1).

	Recall@		Precision@		nDCG@		MRR@	
	5	10	5	10	5	10	5	10
WT10g	0.229	0.454^{134}	0.356	0.394^{1345}	0.382	0.476^{13}	0.590	0.591
Robust	0.231	0.390^{13}	0.385	0.3970^{135}	0.459	0.561^{135}	0.665	0.667^{1345}
TREC8	0.180	0.377^{134}	0.4390	0.470^{1345}	0.464	0.523^{135}	0.653	0.658

Table 4.4: Table showing metric scores from the WT10g dataset for $K = 15$ for the BM25 baseline and the two threshold variants of the DES algorithm, where T is set to 2 and 3. Maximum values are in boldface.

Algorithm	Recall@15	Prec@15	nDCG@15	MRR
DES($T = 3$)	0.325	0.361	0.364	0.419
DES($T = 2$)	**0.358**	**0.391**	**0.447**	**0.637**
BM25 Baseline	0.296	0.329	0.393	0.596

compared to when $M = 10$. When T was set to 3, the DES algorithm generated exploratory rankings for both pages 1 and 2, with λ_1 and λ_2 nominally set to 0.5 and $\lambda_3 = 1$. Document relevance scores were updated after receiving feedback on page 1 and then again after page 2. This was compared against the DES algorithm with the same settings except where $T = 2$ (in this case, the rankings for pages 2 and 3 were found using the *PRP* after observing feedback on page 1), and also the baseline BM25 ranking across 3 pages, giving the results shown in Table (4.4).

The results show that while the $T = 3$ threshold variant still offers some improved results over the baseline across all 3 pages, it performs worse than the case where $T = 2$. This is an example where too much exploration can negatively affect the overall results. When T is set to 3, the DES algorithm creates exploratory rankings in both the first and second pages, before it settles on an optimal ranking for the user on the third page. Except for on particular queries, a user engaging in exploratory search should be able to provide sufficient feedback on the first page in order to optimise the remaining pages, and so setting $T = 2$ is ideal in this situation.

4.5 Conclusion

In this chapter, an algorithm for optimally ranking documents in the multi-page search scenario using the *DIR* framework has been presented. This solution differs from interactive ranking research and also the DIR-MPS algorithm in the previous chapter, by modeling document similarity as a form of document dependence when optimally choosing rankings. By doing this, more effective, exploratory rankings can be chosen for the first results page that explore dissimilar documents, maximizing what can be learned using relevance feedback. This feedback can then be exploited to provide a much improved second page ranking, in a way that is unobtrusive and intuitive to the user. This has been demonstrated to work in a ranking algorithm that utilises Monte-Carlo sampling to make its computation tractable. Using appropriate text collections, demonstrable improvements over a number of similar baselines have been made. Unlike the previous chapter, the baselines in this chapter are practical ranking algorithms rather than framework instantiations. Across three TREC datasets, it was found that some exploration on the first page led to optimal rankings across both pages, and that the DES algorithm out-performed the baselines significantly. Improvements are also demonstrated when using different problem settings such as the number of ranked documents and the number of pages, thus demonstrating the effectiveness of dynamic ranking in multi-page search in a number of different scenarios.

Chapter 5

Dynamic Online Learning to Rank

In this chapter, the *DIR* framework is applied to problems in the area of online learning to rank. Where dynamic multi-page search concerned a user navigating over several pages of dynamically updating search results, in dynamic online learning to rank, an optimal ranking of documents for a single query is found by observing feedback from a population of users over time. The *DIR* framework is used to motivate the derivation of a relevance update function that calculates the posterior probability of a document's relevance as it evolves over time by way of observing user clicks. To this end, a novel click model is proposed that takes into account the rank bias of the clicks and provides a plug-in framework that can incorporate other well-known click models.

Based on the multi-armed bandit theory, the complexity limitations of the *DIR* framework are circumvented by making use of a simple index-based algorithm for implementing the dynamic online learning to rank solution. This algorithm uses a dynamic ranking rule which takes rank bias and the exploration of documents into account. Furthermore, a variant algorithm that uses the portfolio theory of IR to learn an optimally diverse ranking over time is also defined. Experiments using TREC data and simulations investigate the algorithms and the effect that exploration and noise has on their performance. An evaluation is also undertaken using Yandex search log data with mixed results, leading to a discussion on the difficulties of evaluation in *DIR*.

5.1 Introduction

In online learning to rank, an optimal ranking of documents is learned over time by displaying documents to users and learning their preferences through feedback. It is a natural dynamic IR problem, where the dynamic agent is the search system, the population of users is the environment and their clicks the feedback, and the document ranking is the action taken at each stage. Dynamic online learning to rank can also be applied in situations where document content is unclear or incomplete for prior analysis (for instance in collaborative filtering); where new documents are routinely added (such as information filtering); or simply as a complement to existing techniques (such as offline ranking and retrieval).

In this chapter, a dynamic ranking solution for pointwise online learning to rank is proposed. The relevance of individual documents is found by including them in document rankings displayed to a user for a given query. The users' clickthroughs are treated as relevance feedback and used to update the relevance score of each ranked document. The formulation is built upon the *DIR* framework, although the focus here is on the derivation of a suitable relevance update function rather than a dynamic utility. The resulting function takes into account rank position when interpreting clicks for updating document relevance and can flexibly integrate many click models, such as the examination hypothesis and dependent click models [38, 56].

Document independence is a key assumption in this dynamic ranking formulation, but as with the *PRP*, this does not allow for properties such as diversification to be taken into account. To address this, an alternative formulation is proposed that introduces document dependency and diversity using the portfolio theory of IR [40]. Furthermore, the problems of computational complexity encountered for multi-page search are addressed by implementing both formulations as an index-based multi-armed bandit algorithm [118]. This leads to a simple and efficient ranking rule for either exploratively learning an optimal ranking of documents, or else learning a diverse ranking of documents over time.

To evaluate the dynamic online learning to rank algorithms, two types of ex-

periment were conducted. In the first case, users were simulated based on TREC relevance judgement data to verify the effectiveness of the approaches in this chapter and provide some preliminary analysis on the effect of exploration on the performance of various click models. This was followed by an evaluation using the Yandex Relevance Prediction Challenge[1] dataset to test the algorithm in a realistic, practical setting using search log data, to demonstrate its capability and flexibility in learning relevance rankings. The results from this experiment were encouraging but mixed, and indicative of open problems in the area of *DIR* evaluation.

5.1.1 Related Work

Online learning to rank is a currently active area of research in IR. Work that is particularly similar to this include the approach by Xue et al. [156] who used clickthroughs in search logs to infer the similarity between pairs of queries and pairs of documents, and more recently, the work by Shivaswamy, Joachims and Yue [157, 128] who similarly made use of user feedback to improve online learning to rank. Elsewhere, there has been a focus on the application of multi-armed bandits in online learning to rank, particularly in the case where ranked lists are interleaved and clicks are used to learn which is preferential to the user [22], or where search logs were used to improve the learning speed in online learning to rank [158]. A key difference between these approaches and that presented in this chapter is the focus on preference learning (pairwise comparison) or listwise learning as opposed to the pointwise approach taken here, which allows for the inclusion of rank bias, and thus existing click modeling research, when interpreting clicks.

The UCB formulations in this chapter are an extension of the multi-play UCB based MAB [121]. Other multi-play MABs have been developed [120, 159], although the focus has been on theoretical developments (such as improving regret bounds or performance in the adversarial setting) rather than its application to problems in IR.

Related to this chapter's work on diversification over time is the work by Radlinski et al. [23], where a multi-armed bandit algorithm was assigned to each

[1] `http://imat-relpred.yandex.ru/en/`

ranking position and trained to learn the optimal document to rank for it, resulting in diverse rankings. This algorithm is directly compared with the portfolio based approach in this chapter and improvements are demonstrated.

This work differs from the literature in that: 1) a flexible and general click model is demonstrated to accurately interpret the clicks observed on a ranking as feedback, 2) the solution learns pointwise document relevancies rather than listwise or pairwise preferences, and 3) two solutions are derived using the same framework that either rank for document relevance or document diversity.

5.2 Dynamic Relevance Update for Online Learning to Rank

In this problem formulation, the objective is to find an optimal ranking of M documents for a fixed query. For each individual search of the query over a population of users (regarded here as the stage or time step t), a ranking action vector is chosen

$$\vec{\mathbf{a}}_t = \langle d_{t1}, \ldots, d_{tM} \rangle \tag{5.1}$$

where d_{ti} is the document retrieved at rank i and at time t (where $d_{ti} \neq d_{tj} \; \forall \; i \neq j$).

The static utility of a particular time step can be set as $U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \sum_{i=1}^{M} R_{ti}$, where R_{ti} is the hidden, binary relevance of document d_{ti}. This simple utility function is maximised when all of the documents in the ranking $\vec{\mathbf{a}}_t$ are relevant. Given that R cannot be directly observed, instead an assumption is made that the observable clickthroughs for each ranking are explicit, binary relevance labels, and so the static utility can be redefined as

$$U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) = \sum_{i=1}^{M} o_{ti} \tag{5.2}$$

where o_{ti} is the click observation of document d_{ti}.

Thus, the goal in dynamic online learning to rank is to find the optimal rank action $\vec{\mathbf{a}}^*$ that maximises the users' static utility over time, i.e. their expected number

of clicks, given by

$$\vec{\mathbf{a}}^* = \langle \vec{\mathbf{a}}_1^*, \ldots, \vec{\mathbf{a}}_T^* \rangle = \underset{\langle \vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_T \rangle}{\operatorname{argmax}} \sum_{t=1}^{T} \mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t)\big] \tag{5.3}$$

5.2.1 Mixed Click Model

According to Eq. (3.18) in the *DIR* framework, the observation vector $\vec{\mathbf{o}}$ for a ranking can be marginalised over in the expected utility of the objective function in Eq. (5.3), allowing it to be expressed as

$$\sum_{t=1}^{T} \mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t)\big] = \sum_{t=1}^{T} \sum_{\vec{\mathbf{o}} \in \mathscr{O}} P(\vec{\mathbf{o}}_t = \vec{\mathbf{o}} | \vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) \tag{5.4}$$

$$= \sum_{t=1}^{T} \sum_{\vec{\mathbf{o}} \in \mathscr{O}} P(\vec{\mathbf{o}}_t = \vec{\mathbf{o}} | \vec{\mathbf{a}}_t, \vec{\mathbf{r}}_t) \sum_{i=1}^{M} o_{ti} \tag{5.5}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{M} \sum_{o \in \{0,1\}} P(o_{ti} = o | d_{ti}, r_{ti}) o_{ti} \tag{5.6}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{M} P(o_{ti} = 1 | d_{ti}, r_{ti}) \tag{5.7}$$

Here, Eq. (5.6) is a result of assuming that the clickthroughs are independently and identically distributed across users for each time step. A related assumption is that the relevance of documents is also independent of other documents, similar to the *PRP*. For this formulation, the first assumption is reasonable as clickthroughs are observed over a population of unrelated users who can be assumed to be independent of one another. Ranking under the assumption of document independence has already been discussed in this thesis in Section 3.2.1.2 and is further addressed in the alternative problem formulation in Section 5.3. It is worth noting that click observations may still be dependent on other observed clicks in the same document ranking.

The observation likelihood function given by the probability distribution in Eq. (5.7) can be calculated using a click model, as was the case with the examination hypothesis click model in Eq. (3.25) in Chapter 3. In this chapter, a novel click model is proposed instead. Here, an assumption is made that the probability of a

click observation can be modeled using a mixture of two binomial distributions, one for rank bias and one for relevance. A hidden binary variable G is used to represent the membership, i.e. $G_i = 0$ if a click occurs due to the rank bias at rank i, and $G_i = 1$ if it occurs due to the document relevance. This results in the following conditional probability of the click (the time step variable t is implied when omitted):

$$P(o_i|d_i,r_i) = P(o_i|d_i,r_i,G_i=1)P(G_i=1) + P(o_i|G_i=0)P(G_i=0) \tag{5.8}$$

$$= P(o_i|d_i,r_i)P(G_i=1) + P(o_i|G_i=0)P(G_i=0) \tag{5.9}$$

$$= \hat{r}_i^{o_i}(1-\hat{r}_i)^{1-o_i}g_i + b_i^{o_i}(1-b_i)^{1-o_i}(1-g_i) \tag{5.10}$$

Here, three parameters have been defined:

$$\hat{r}_i \equiv P(o_i = 1|d_i,r_i) \tag{5.11}$$

$$b_i \equiv P(o_i = 1|G_i = 0) \tag{5.12}$$

$$g_i \equiv P(G_i = 1) \tag{5.13}$$

where $\hat{r}_i$ is the estimated probability of relevance for document d_i, b_i is the rank bias at position i and g_i is the probability that a user will click due to the document's relevance rather than due to the rank i.e. it could be considered as a measure of the *trust* that the user has in the search system. It is worth noting that the mixed click model here is a generalisation of the mixture model evaluated by Craswell et al. [38]. Likewise, the heuristic background click model by Agichtein et al. [160] is in fact the case where $g_i = 1$ and b_i is considered as the background click rate. It is shown later in this chapter that other click models such as the examination hypothesis [38] and dependent click model [56] are also special cases which can be plugged into this model by the setting of the parameters in Eq. (5.11), Eq. (5.12) and Eq. (5.13).

5.2.2 Expectation Maximisation

The mixed click model described in Eq. (5.10) is made up of three parameters, $\hat{r}_i$, b_i and a hidden parameter g_i (for a given rank position i). Estimates can be made

for the optimal settings for each parameter based on the click observations by using the Expectation Maximization (EM) algorithm [161]. In the Expectation (E) step, an appropriate setting for the hidden random variable G_{Ti} at time step T is found based on the observation at that time step o_{Ti}, given as:

E Step:

$$P(G_{Ti}|o_{Ti}) = \frac{P(o_{Ti}|G_{Ti})P(G_{Ti})}{P(o_{Ti}|G_{Ti}=1)P(G_{Ti}=1)+P(o_{Ti}|G_{Ti}=0)P(G_{Ti}=0)} \tag{5.14}$$

which is derived by applying Bayes Rule.

The Maximisation (M) step involves finding the best parameter settings for $\hat{r}_i$, b_i and g_i that maximise the likelihood of all of the observations at a particular rank $\vec{\mathbf{o}}_i = \langle o_{1i}, \ldots, o_{Ti} \rangle$ up until step T. This likelihood function is given by:

M Step:

$$\begin{aligned}
L(\hat{r}_i, b_i, g_i|\vec{\mathbf{o}}_i) &= \prod_{t=1}^{T} P(o_{ti}, |d_i, r_i) && (5.15)\\
&= \prod_{t=1}^{T} \sum_{G_i \in \{0,1\}} P(o_{ti}, G_i|d_i, r_i) && (5.16)\\
&\propto \sum_{t=1}^{T} \log \left[\sum_{G_i \in \{0,1\}} P(o_{ti}, G_i|d_i, r_i) \right] && (5.17)\\
&= \sum_{t=1}^{T} \log \left[\sum_{G_i \in \{0,1\}} P(G_{ti}|o_{ti}) \frac{P(o_{ti}, G_i|d_i, r_i)}{P(G_{ti}|o_{ti})} \right] && (5.18)\\
&\geq \sum_{t=1}^{T} \sum_{G_i \in \{0,1\}} P(G_{ti}|o_{ti}) \log \left[\frac{P(o_{ti}, G_i|d_i, r_i)}{P(G_{ti}|o_{ti})} \right] && (5.19)
\end{aligned}$$

where the lower bound in Eq. (5.19) follows from Jensen's inequality [161]. The probability distribution $P(o_{ti}, G_i|d_i, r_i)$ is simply the probability of a click due to either the relevance or bias setting of the mixed click model in Eq. (5.10), and can be simplified to:

$$P(o_{ti}, G_i = 1|d_i, r_i) = \hat{r}_i^{o_i}(1-\hat{r}_i)^{1-o_i} g_i \tag{5.20}$$

$$P(o_{ti}, G_i = 0|d_i, r_i) = b_i^{o_i}(1-b_i)^{1-o_i}(1-g_i) \tag{5.21}$$

giving the log-likelihood lower bound function in Eq. (5.19) as

$$l(\hat{r}_i, b_i, g_i|\vec{\mathbf{o}}_i) = \sum_{t=1}^{T} P(G_{ti}=1|o_{ti}) \log\left[\frac{\hat{r}_i^{o_i}(1-\hat{r}_i)^{1-o_i} g_i}{P(G_{ti}=1|o_{ti})}\right] + P(G_{ti}=0|o_{ti}) \log\left[\frac{b_i^{o_i}(1-b_i)^{1-o_i}(1-g_i)}{P(G_{ti}=0|o_{ti})}\right] \tag{5.22}$$

To derive the M step for each parameter, derivatives of Eq. (5.22) can be taken in order to find the maximum value of each parameter, giving the update function

$$\frac{\partial l}{\partial g_i} = \sum_{t=1}^{T} \frac{P(G_{ti}=1|o_{ti})}{g_i} + \frac{P(G_{ti}=0|o_{ti})}{g_i - 1} = 0 \tag{5.23}$$

$$\implies g_i = \sum_{t=1}^{T} \frac{P(G_{ti}=1|o_{ti})}{P(G_{ti}=1|o_{ti}) + P(G_{ti}=0|o_{ti})} \tag{5.24}$$

$$\implies g_i = \frac{1}{T} \sum_{t=1}^{T} P(G_{ti}=1|o_{ti}) \tag{5.25}$$

for the mixed model parameter, likewise the update function for the document relevance estimate is

$$\frac{\partial l}{\partial \hat{r}_i} = \sum_{t=1}^{T} \frac{P(G_{ti}=1|o_{ti})(\hat{r}_i - o_{ti})}{(\hat{r}_i - 1)\hat{r}_i} = 0 \tag{5.26}$$

$$\implies \hat{r}_i = \frac{\sum_{t=1}^{T} o_{ti} \times P(G_{ti}=1|o_{ti})}{\sum_{t=1}^{T} P(G_{ti}=1|o_{ti})} \tag{5.27}$$

and through a similar derivation, the update function for the bias parameter is

$$b_i = \frac{\sum_{t=1}^{T} o_{ti} \times P(G_{ti}=0|o_{ti})}{\sum_{t=1}^{T} P(G_{ti}=0|o_{ti})} \tag{5.28}$$

5.2.3 Relevance Update Function

The parameter update functions in Eq. (5.25), Eq. (5.27) and Eq. (5.28) can be solved at every time step, although in this formulation, the settings for g_i and b_i are fixed. Not only does this simplify the approach, but it allows for different click models to be set in advance. Following this, the relevance update function can be

simplified by defining

$$\alpha_{ti} \equiv P(G_{ti}=1|o_{ti}=1) \tag{5.29}$$
$$= \frac{P(o_{ti}=1|G_{ti}=1)P(G_{ti}=1)}{P(o_{ti}=1|G_{ti}=1)P(G_{ti}=1)+P(o_{ti}=1|G_{ti}=0)P(G_{ti}=0)} \tag{5.30}$$
$$= \frac{\hat{r}_{ti}g_i}{\hat{r}_{ti}g_i+b_i(1-g_i)} \tag{5.31}$$

and

$$\beta_{ti} \equiv P(G_{ti}=1|o_{ti}=0) \tag{5.32}$$
$$= \frac{P(o_{ti}=0|G_{ti}=1)P(G_{ti}=1)}{P(o_{ti}=0|G_{ti}=1)P(G_{ti}=1)+P(o_{ti}=0|G_{ti}=0)P(G_{ti}=0)} \tag{5.33}$$
$$= \frac{(1-\hat{r}_{ti})g_i}{(1-\hat{r}_{ti})g_i+(1-b_i)(1-g_i)} \tag{5.34}$$

where $\hat{r}_{ti}$ is the probability of relevance estimated from all past click observations using Eq. (5.27). With α_{ti} and β_{ti}, the probability of relevance update function calculated at time T can be simplified to:

$$\hat{r}_{Ti} = \frac{\sum_{t=1}^{T} o_{ti}\alpha_{ti}^{o_{ti}}\beta_{ti}^{1-o_{ti}}}{\sum_{t=1}^{T} \alpha_{ti}^{o_{ti}}\beta_{ti}^{1-o_{ti}}} \tag{5.35}$$

This can be updated recursively over time to give the *DIR* relevance update function τ, which in this scenario is

$$\hat{r}_{Ti} = \hat{r}_{T-1,i}\frac{\gamma_{T-1,i}}{\gamma_{Ti}} + o_{Ti}\left(1-\frac{\gamma_{T-1,i}}{\gamma_{Ti}}\right) \tag{5.36}$$

where

$$\gamma_{ti} \equiv \sum_{k=1}^{t} \alpha_{ki}^{o_{ki}}\beta_{ki}^{1-o_{ki}} \tag{5.37}$$

can be interpreted as the "effective" number of impressions, a utility representing the number of times a document is likely to have been examined by a user based on

its rank bias and probability of relevance.

5.2.3.1 Analysis of α and β

α_{ti} and β_{ti} can be considered as the "effective" number of observations of clicks and non-clicks respectively, which is based on the rank position of the observation. For instance, the value of α_{ti} increases with rank position i; it is larger if a click is observed on a document ranked at the bottom of an SERP, which is unlikely to have been clicked purely due to rank bias, thus the observation is a very strong signal of relevance. In contrast, if a click is observed on a top ranking document, then the corresponding α count is small as the click is more likely to be due to rank bias.

Conversely, β_{ti} decreases with respect to i i.e. if a top ranked document has not been clicked then the value for β is large, otherwise its value is small for non-clicks observed on lower ranking documents. In effect, this penalises documents at high ranking positions that do not receive clicks, a strong signal that the document is incorrectly ranked. Likewise, a non-click occurring on a low-ranked document is expected due to the rank bias, and so penalised less.

In summary, a lack of click observations on high-ranking documents or clicks occurring on low-ranked documents are both important observation signals that carry more weight than the converse, and so have a larger effect on the overall "effective" number of impressions γ which is used to updated the estimated probability of relevance.

5.2.4 Dynamic Utility

Based on the derivations so far, the objective function in Eq. (5.3) can be restated as the dynamic utility function

$$U_D(\vec{\mathbf{r}}, T) = \sum_{t=1}^{T} \sum_{i=1}^{M} P(o_{ti} = 1 | d_{ti}, r_{ti}) \tag{5.38}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{M} \left[\hat{r}_{ti} g_i + b_i (1 - g_i) \right] \tag{5.39}$$

where the objective is to find the optimal sequence of rank actions $\vec{\mathbf{a}}^* = \text{argmax}_{\langle \vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_T \rangle} U_D$, subject to the relevance update in Eq. (5.36).

As mentioned, the setting of the fixed parameters g_i and b_i can allow different click models to be incorporated into the dynamic online learning to rank solution. For instance, this is demonstrated in the experiments in Section 5.4, where the examination hypothesis and dependent click models are plugged into the algorithm and evaluated alongside the mixed click model. The general parameter settings for each click model are:

Mixed Clicks $g_i,\ b_i$

Examination Hypothesis $g_i,\ b_i \equiv 0$, where the mixed click parameter g can be considered the rank bias

Dependent Click Model $g_i \equiv \prod_{j=1}^{i-1}(1-\hat{r}_j+b_j\hat{r}_j),\ b_i \equiv 0$, where the mixed click parameter at rank i is weighted by the relevance of previously ranked documents.

5.2.5 Dynamic Ranking (UCB-DR) Algorithm

In this dynamic online learning to rank formulation, the dynamic utility is constrained by the relevance update function in Eq. (5.36), which updates the probability of relevance estimate for each document after it has been displayed to the user and a subsequent click observation made. In previous chapters, the dynamic utility has been directly solved in order to determine the optimal ranking action, albeit with suitable approximations in order to make the calculation tractable.

In the **Upper Confidence Bound - Dynamic Ranking (UCB-DR)** algorithm in Algorithm 4, the UCB multi-armed bandit algorithm is used as the basis for a simple, index-based ranking scheme. Here, an index score Λ_d is assigned to each document that is based on its estimated probability of relevance. If a document is included in the rank action at time t, a click observation on that document can be made and its probability of relevance updated accordingly using Eq. (5.36). Through this, the relevance of documents can be learned and those with the highest probability of relevance are *exploited* by being displayed in the top ranking positions.

On the other hand, the *exploration* of documents is achieved by supplementing the probability of relevance with an upper confidence bound value in Λ. This

Algorithm 4 The UCB-DR Algorithm

function UCB-DR($\hat{\vec{\mathbf{r}}}, \vec{\mathbf{b}}, \vec{\mathbf{g}}, \lambda$)
  $\gamma_d = 1 \; \forall d \in D$ ▷ Initialize effective impressions
  loop $t \leftarrow 1$ to T
    for all $d \in D$ **do**
      $\Lambda_d = \hat{r}_d + \lambda \times \sqrt{\frac{2 \ln t}{\gamma_d}}$ ▷ Index score
    end for
    $\vec{\mathbf{a}}_{1 \to M} = \text{SORT}(\Lambda, \text{descend})[1 \to M]$
    Display $\vec{\mathbf{a}}$ to user
    Retrieve observation vector of clicks $\vec{\mathbf{o}}$
    loop $i \leftarrow 1$ to M ▷ For each rank
      $d = d_i$ ▷ The document ranked at position i
      $\alpha_i = \frac{\hat{r}_d g_i}{\hat{r}_d g_i + b_i(1-g_i)}$ ▷ Eq. (5.31)
      $\beta_i = \frac{(1-\hat{r}_d) g_i}{(1-\hat{r}_d) g_i + (1-b_i)(1-g_i)}$ ▷ Eq. (5.34)
      $\hat{\gamma}_d = \gamma_d + \alpha_i^{o_i} \beta_i^{1-o_i}$ ▷ Eq. (5.37)
      $\hat{r}_d = \hat{r}_d \frac{\gamma_d}{\hat{\gamma}_d} + o_i \left(1 - \frac{\gamma_d}{\hat{\gamma}_d}\right)$ ▷ Eq. (5.36)
      $\gamma_d = \hat{\gamma}_d$
    end loop
  end loop
end function

bound grows larger with each time step t, causing the Λ value to sufficiently increase enough to trigger the exploration of those documents with a lower probability of relevance, if they have not been shown to the user for some time. Conversely, when a document is displayed to the user, then the "effective" number of impressions γ increases depending on the click observation. Large increases in γ indicate that more information has been learned about the document relevance, and so the upper confidence bound decreases accordingly. As a result, documents that are displayed at the bottom of the SERP are not unjustly penalised for not being clicked on, and can be promoted to a higher rank where there is a greater chance of observing a click.

The exploration parameter λ can be tuned to optimise the amount of exploration that occurs when generating a document ranking. When set to 0, the UCB-DR algorithm acts myopically and greedily ranks documents in a strictly decreasing order of relevance according to the *PRP*.

5.3 Online Diversification

In the previous section, an optimal ranking of documents was found over time by displaying different rankings of documents to users. An assumption made early in the formulation was the independence of documents. In this section, the focus is changed to finding an optimally *diverse* ranking of documents subject to the assumption that documents are dependent on one another.

Once again the static utility $U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})$ is set as the number of clicks given in Eq. (5.2). In order to introduce dependence, the exponential utility $U_e(x) = 1 - \exp(-\lambda x)$, one of the commonly used risk averse utilities, is applied to the static utility. Here, $\lambda$is a predefined utility parameter that is used to adjust the risk preference. By also making the assumption that the clicks have normally distributed noise (a similar assumption is made with the setting of the multi-variate Gaussian distribution in the previous chapters), the overall dynamic utility becomes

$$U_D(\vec{\mathbf{r}}, T) = \sum_{t=1}^{T} \mathrm{E}\left[U_e\big(U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big)\right] \tag{5.40}$$

$$= \sum_{t=1}^{T} \mathrm{E}\left[1 - \exp\big(-\lambda U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big)\right] \tag{5.41}$$

$$= \sum_{t=1}^{T} \left(1 - \exp\big(-\lambda \mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] + \frac{\lambda^2}{2}\mathrm{Var}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big]\big)\right) \tag{5.42}$$

where Eq. (5.42) follows from the Gaussian noise assumption. Thus, the objective is to find the optimal ranking sequence

$$\vec{\mathbf{a}}^* = \underset{\langle \vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_T \rangle}{\operatorname{argmax}} \sum_{t=1}^{T} \mathrm{E}\left[U_e\big(U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big)\right] \tag{5.43}$$

$$= \underset{\langle \vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_T \rangle}{\operatorname{argmax}} \sum_{t=1}^{T} \left(1 - \exp\big(-\lambda \mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] + \frac{\lambda^2}{2}\mathrm{Var}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big]\big)\right) \tag{5.44}$$

$$= \underset{\langle \vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_T \rangle}{\operatorname{argmax}} \sum_{t=1}^{T} \big(1 - \exp(-\lambda\theta)\big) \tag{5.45}$$

where $\theta = \mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] - \frac{\lambda}{2}\mathrm{Var}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big]$. Thus, the maximisation of Eq. (5.45) over $\vec{\mathbf{a}}^*$ is equivalent to maximising θ, giving the overall objective

function

$$\vec{\mathbf{a}}^* = \underset{\langle \vec{\mathbf{a}}_1,\ldots,\vec{\mathbf{a}}_T \rangle}{\operatorname{argmax}} \sum_{t=1}^{T} \left(\mathrm{E}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] - \frac{\lambda}{2} \mathrm{Var}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] \right) \tag{5.46}$$

Here, the objective has been decomposed into two parts; the exploitative expected number of clicks that is maximised when relevant documents are chosen for $\vec{\mathbf{a}}$, and the explorative variance which is shown in the next section to encourage document diversity. As with the UCB-DR algorithm, the parameter λ trades off exploration and exploitation, in this case, how much diversification occurs in the search results.

5.3.1 Correlation and Co-Clicks

The variance in Eq. (5.46) can be decomposed into

$$\begin{aligned}
\mathrm{Var}\big[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\big] &= \mathrm{Var}\left[\sum_{i=1}^{M} o_{ti}\right] && (5.47)\\
&= \sum_{i=1}^{M} \mathrm{Var}[o_{ti}] + \sum_{i \neq j}^{M} \mathrm{Cov}[o_{ti}, o_{tj}] && (5.48)\\
&= \sum_{i=1}^{M} \mathrm{Var}[o_{ti}] + 2\sum_{i=1}^{M} \sum_{j=i+1}^{M} \mathrm{Cov}[o_{ti}, o_{tj}] && (5.49)\\
&= \sum_{i=1}^{M} \mathrm{Var}[o_{ti}] + 2\sum_{i=1}^{M} \sum_{j=i+1}^{M} \sigma_{ti}\sigma_{tj}\rho(o_{ti}, o_{tj}) && (5.50)
\end{aligned}$$

where σ_i is the standard deviation of the observation of the document ranked at position i. $\rho(o_i, o_j)$ is the correlation coefficient between the clicks for the documents ranked at positions i and j, where $\rho \in [-1, 1]$.

The correlation coefficient is a measure of the type of dependency between two documents. When $\rho = 1$, the two documents are positively correlated with one another, meaning that a click on one of the documents is likely to coincide with a click on the other. Conversely, when $\rho = -1$, the documents are negatively correlated, and so a click on one document is likely to coincide with a non-click on the other. When $\rho = 0$, the documents are independent.

An estimation of the correlation coefficient between pairs of documents can be made by measuring the ratio of **co-clicks** to **co-shows**. The number of co-clicks $X(i,j)$ between documents d_i and d_j is increased by 1 whenever a click occurs on both documents in the same ranking, otherwise it is decreased by 1 whenever a click only occurs on one of the documents. If no clicks are observed on either, or the two documents do not appear in the same ranking, then the value remains the same. Likewise, the number of co-shows $Y(i,j)$ increases by 1 whenever the two documents are displayed in the same ranking together, otherwise it does not change. $X(i,i) = X(i)$ and $Y(i,i) = Y(i)$ record respectively the number of clicks and displays for a single document.

Thus, the estimated correlation is $\hat{\rho}(i,j) = \frac{X(i,j)}{Y(i,j)}$. It is worth noting that when documents are independent, there is the risk of introducing systematic error into the correlation estimate. For example, two independent documents with a probability of relevance $r_i = r_j = 0.5$ will have a correlation of $-\frac{1}{3}$ rather than 0. In this formulation, all documents are assumed to have some dependence on one another.

5.3.2 Portfolio-armed Bandit (PAB) Algorithm

The variance of the static utility is decomposed into two components in Eq. (5.50), the variance of the clicks on each document, and also the correlation coefficient. The variance can be likened to the upper-confidence bound in the UCB algorithm, representing the uncertainty of the probability of relevance estimate. Also, the positive or negative correlation can be considered as a measure of whether documents belong to the same subtopic, which is made explicit in the simulation in Section 5.4.3.

The dynamic utility can now be redefined as

$$U_D(\vec{\mathbf{r}},T) = \sum_{t=1}^{T}\left(\mathrm{E}\big[U_S(\vec{\mathbf{a}}_t,\vec{\mathbf{r}})\big] - \frac{\lambda}{2}\sum_{i=1}^{M}\mathrm{Var}[o_{ti}] - \lambda\sum_{i=1}^{M}\sum_{j=i+1}^{M}\rho(o_{ti},o_{tj})\right) \tag{5.51}$$

In this online diversification formulation, the relevance of a document depends on whether it belongs to the user's subtopic preference. Furthermore, rank bias is assumed not to affect the probability of a click, which is a simplifying assumption that means that the mixed click model is not needed. As a result, the expected

Algorithm 5 The PAB Algorithm

function PAB(λ)
  $\mathbf{X} = 1, \mathbf{Y} = 1$ ▷ Initialize all co-clicks and co-shows to 1
  loop $t \leftarrow 1$ to T
    for all $d \in D$ **do**
      $\Lambda_d = \frac{X(d)}{Y(d)} + \sqrt{\frac{2 \ln t}{Y(d)}}$ ▷ Index score
    end for
    $\vec{\mathbf{a}} = \text{ARRAY}(M)$
    $a_1 = \text{argmax}_d \Lambda_d$
    loop $i \leftarrow 2$ to M ▷ Sequential ranking decision
      $a_i = \text{argmax}_{d \notin \vec{\mathbf{a}}} \left\{ \Lambda_d - \lambda \sum_{j=1}^{i-1} \frac{X(d,d_j)}{Y(d,d_j)} \right\}$ ▷ Portfolio theory of IR
    end loop
    Display $\vec{\mathbf{a}}$ to user
    Retrieve observation vector of clicks $\vec{\mathbf{o}}$
    loop $i \leftarrow 1$ to M
      loop $j \leftarrow 1$ to M
        $X(i,j) = X(i,j) + o_i o_j - (o_i - o_j)^2$ ▷ Update co-clicks
        $Y(i,j) = Y(i,j) + o_i + o_j - o_i o_j$ ▷ Update co-shows
      end loop
    end loop
  end loop
end function

static utility can simply be set to $\mathrm{E}\left[U_S(\vec{\mathbf{a}}_t, \vec{\mathbf{r}})\right] = \sum_{i=1}^{M} \hat{r}_{ti}$, and $\hat{r}_{ti}$ approximated by the maximum likelihood estimate $\frac{X(d_{ti})}{Y(d_{ti})}$.

Eq. (5.51) has the same structure as the objective function in Eq. (2.8) for the portfolio theory of IR [40]. In portfolio theory ranking, the variance in the objective function is minimised by choosing a diverse ranking i.e. documents that are negatively correlated with one another. This is incorporated into the **Portfolio-Armed Bandit (PAB)** algorithm in Algorithm 5, which is designed to learn document correlations and create a diverse ranking of documents over time.

The PAB algorithm is a variation of the UCB-DR algorithm and features similar features, such as a UCB index value Λ which is used to choose relevant documents to rank and balances exploration and exploitation through the upper confidence bound. Here, the variance value in Eq. (5.51) is *added* rather than subtracted so that exploration can occur. On the other hand, the portfolio theory of IR *subtracts* the document correlations with already ranked results by way of a sequential

ranking decision, through which the diversification of the results occurs. The exploration parameter λ is used to balance the effect of the explorative probability of relevance given by Λ and the diversification caused by the portfolio theory of IR.

5.4 Experiments

In this chapter, two online learning to rank algorithms have been proposed, the UCB-DR algorithm which learns an optimal ranking of documents over time and includes a click model for interpreting click observations at different rank positions, and the PAB algorithm that learns document correlations and a diverse ranking of documents over time. The ideal evaluation of an online learning algorithm would be to present the dynamically generated rankings to users of an operational search engine and to observe their interaction behaviour. Having no access to such a system, instead each of the algorithms is evaluated separately using a combination of simulations and search logs.

5.4.1 UCB-DR Simulation Analysis

In this experiment, the UCB-DR algorithm was evaluated using user simulations based on a TREC dataset to 1) show that the algorithm does converge to an optimal ranking that can maximise evaluation metrics such as MAP and nDCG over time, and 2) determine the benefit of exploration by contrasting UCB-DR with a myopic variant, and subsequently find an ideal value for λ.

Three click model based variants of the UCB-DR algorithm were evaluated: the Mixed Click (denoted as *UCB-DR-MC*), Examination Hypothesis (*UCB-DR-EH*) and Dependent Click (*UCB-DR-DC*) models. These models were chosen as they are well-regarded in the academic community, they have been thoroughly studied and their effectiveness verified using search log data [38], and they were easily plugged into the mixed click model introduced in this chapter.

5.4.1.1 TREC-Based Simulation

To simulate a realistic collection of documents and their associated relevance values, relevance judgements from the TREC10 Web Ad Hoc Retrieval Track were used. The UCB-DR variant algorithms were run over 50 different topics (representing

50 queries), each of which contained an average of 1408 relevance judgements with 41.53 documents judged as relevant. Judgements were graded either 0, 1 or 2, which were normalised to give probability of relevance values 0, $\frac{1}{2}$ and 1.

At each time step, a ranking of $M = 10$ documents was generated by each of the UCB-DR variants. A simulated user then examined and 'clicked' on each of the documents. This observation vector of clicks and non-clicks was determined stochastically using the click model of the specific UCB-DR variant i.e. clicks were generated according to the examination hypothesis model when the UCB-DR-EH algorithm was being tested. With no data from which to learn the click model parameters, the following settings were used (the value of 0.8 was picked as it delivered consistent performance across all click models and topics):

Mixed Click Model $g_i \equiv 0.8 \quad b_i \equiv 0.8^{i-1}$

Examination Hypothesis $g_i \equiv 0.8^{i-1} \quad b_i \equiv 0$

Dependent Click Model $g_i \equiv \prod_{j=1}^{i-1}(1-\hat{r}_j+0.8\hat{r}_j) \quad b_i \equiv 0$ (updated at each time step)

The underlying TREC relevance judgements were used to evaluate the MAP and nDCG@10 metrics for the ranking generated at the end of the learning threshold T, which was set to 500 for each topic. The evaluation was repeated 100 times to find the average performance over the simulated users. Different settings for the exploration parameter λ were used, and the results averaged over all test topics are given in Table (5.1).

From the table it can be seen that the exploration parameter λ is essential for tuning the model as the performance for $\lambda = 1$ is consistently poor. Conversely, it can also be seen that in the myopic case (where $\lambda = 0$) the performance is very good, indicating that even without exploration, the UCB-DR algorithm is able to accurately learn document relevance over time and generate rankings that give good MAP and nDCG@10 scores across all click models. As λ varies from 1 to 0, an improvement in metric scores is observed which can sometimes be better than the myopic case. This suggests that for larger values of λ, the model suffers from

Table 5.1: Average MAP and nDCG@10 scores after $T = 500$ time steps for each UCB-DR variant and for each value of the exploration parameter λ. Maximum values for each variant are in boldface.

MAP								
λ	0	0.001	0.01	0.05	0.1	0.2	0.5	1
MC	0.833	0.834	0.839	0.843	0.859	**0.885**	0.489	0.111
EH	0.798	0.802	0.808	0.839	**0.863**	0.861	0.188	0.093
DCM	0.811	0.802	0.812	0.844	**0.860**	0.858	0.827	0.671
nDCG@10								
MC	**0.998**	0.998	0.998	0.997	0.991	0.977	0.614	0.236
EH	0.994	**0.997**	0.994	0.993	0.979	0.933	0.339	0.207
DCM	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	0.987	0.946	0.806

too much exploration and doesn't generate exploitative rankings, whereas there exists an optimal, small amount of exploration that produces better rankings than the purely myopic case. This result mirrors the findings from Chapter 4, where some exploration was also found to be beneficial in the multi-page setting.

The nDCG@10 scores proved less discriminatory as the algorithm was often able to converge on the optimal ranking action after T steps, whereas the MAP score also reflected the number of relevant documents not ranked. As such, this metric helped choose the parameter setting $\lambda = 0.1$ used in the search log based evaluation in the next section.

5.4.2 UCB-DR Yandex Search Log Experiment

Following the encouraging simulation results, in the next experiment a search log dataset was used to demonstrate 1) that search logs could be used to estimate document relevancies in advance of online learning to rank, 2) the realistic setting of click model parameters, and 3) how the log could be used to evaluate the performance of the UCB-DR algorithms and to provide an upper bound to compare against.

Here, the Yandex Relevance Prediction Challenge[2] dataset was used, an anonymised search log containing 43,977,859 search impressions, each containing

[2] `http://imat-relpred.yandex.ru/en/`

a query, the documents displayed at ranks 1 to 10 and any clicks on those documents. No additional document feature or content information was included, making it well suited to this chapter's *DIR* formulation that learns through user feedback alone. In addition, the log contained 71,930 relevance judgements for training purposes. In this experiment, the 30,717,251 unique queries were narrowed down to 1,327 queries that were searched for in over 1000 impressions and contained at least 10 relevance judgements.

A similar evaluation to the previous TREC simulation was performed, where different click model variants of the UCB-DR algorithm generated rankings of $M = 10$ documents. The analysis was restricted to the two settings $\lambda = 0$ and $\lambda = 0.1$, so that the effect of exploration could be compared against the myopic ranking. Relevance judgements were used to evaluate MAP and nDCG@10 for the ranking at the threshold time step T i.e. the last occurrence of the query in the dataset.

In order to evaluate the ability of the algorithm to learn from existing search log data, a training phase was also introduced, where the first 50% of the occurrences of the query in the search log were used purely to calculate the probability of relevance estimates for the documents, and the remaining 50% were tested over.

5.4.2.1 Interpreting Clickthroughs for Evaluation

A problem inherent to evaluation in the *DIR* framework is the need to observe feedback so that the relevance model can be updated and improved over time. In this thesis, this has been avoided by considering short time thresholds, using relevance judgements as feedback and simulating user clicks. In this experiment, the clicks from the search log were used.

The benefit of this approach was that the clicks represented actual observations of user behaviour when they were interacting with the original search results. The drawback was that these observations only exist for the documents that were originally displayed to the user. In the course of running the UCB-DR algorithm over the queries in the Yandex dataset, documents were chosen to be ranked by UCB-DR that were not shown in the original ranking. In such cases, feedback could not be observed for that document and as a result it would subsequently be penalised by

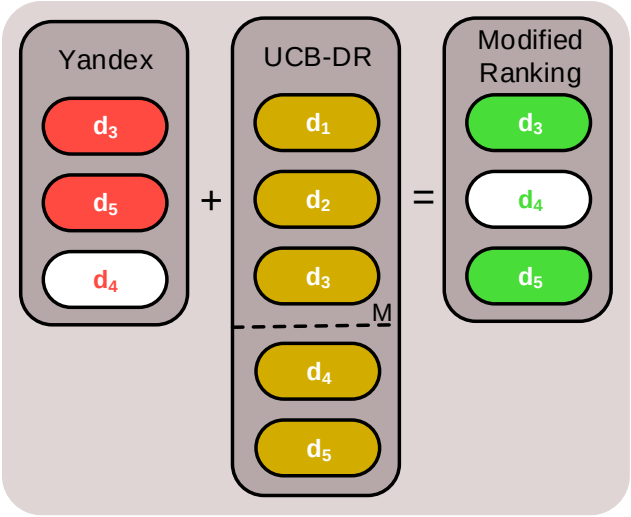


Figure 5.1: Example demonstrating the method used to overcome the restrictions of the Yandex dataset. $d_1 \rightarrow d_5$ are documents chosen by UCB-DR to display at rank positions $1 \rightarrow 5$. Only a subset of these documents are found in the ranking in the Yandex dataset, which is reflected in the modified ranking. Further to this, there is a clicked document in the Yandex dataset (d_4 and shown in white), which is also interpreted as a click in the modified ranking, although at the original rank position of 3 in the Yandex dataset.

the ranking algorithm.

To counter this, in this evaluation the UCB-DR algorithm was limited to only being able to choose documents that were already ranked for the t'th impression, instead re-ranking them according to its estimate of the probability of relevance. In this way, the limitation of not being able to observe feedback for a ranked document was removed. In addition, when updating the probability of relevance, the clickthroughs and rank positions used in the update function were those found in the search logs, not from the ranking generated by UCB-DR. This is illustrated in Fig. (5.1), where the third column represents the 'modified' re-ranking that would be generated based on the ranking in the search log and that generated by UCB-DR.

Despite these changes, the following limitations still existed: 1) the UCB-DR algorithm could only perform as well the ranking in the search log. This was because the algorithm could not discover new relevant documents and receive user feedback on them, stifling exploration. Thus, the performance of the ranking in the dataset itself is considered as an upper bound, and 2) using the click positions from the dataset meant that the observations are used in an unbiased way, but also

Table 5.2: MAP and nDCG@10 scores ± 95% confidence intervals for each UCB-DR click model variant, for both the explorative ($\lambda = 0.1$) and myopic cases ($\lambda = 0$), and for the 0% and 50% training variants. These scores are the averaged scores for the ranking generated after the final impression for each query in the dataset. Maximal UCB-DR scores are in boldface.

MAP (0% training phase)				
λ	Yandex	**MC**	**EH**	**DC**
0.1	0.701 ± .056	0.566 ± .061	0.537 ± .062	0.632 ± .061
0		0.600 ± .063	0.571 ± .064	**0.637 ± .061**
nDCG@10 (0% training phase)				
0.1	0.833 ± .046	0.725 ± .058	0.691 ± .057	0.778 ± .056
0		0.747 ± .057	0.721 ± .056	**0.783 ± .056**
MAP (50% training phase)				
0.1	0.701 ± .056	0.636 ± .044	0.618 ± .046	0.658 ± .046
0		0.652 ± .045	0.635 ± .046	**0.661 ± .047**
nDCG@10 (50% training phase)				
0.1	0.833 ± .046	0.725 ± .058	0.691 ± .057	0.776 ± .056
0		0.747 ± .057	0.721 ± .056	**0.780 ± .056**

resulted in a slow learning rate for the UCB-DR algorithm

5.4.2.2 Experiment and Analysis

Before running each experiment, the search log data and relevance judgements were used to estimate ideal parameter settings for each of the click models. The values for g_i and b_i were calculated at each rank and for each query using maximum likelihood estimators i.e. the number of occurrences of clicks at each rank i for relevant documents etc. In this way, there were no prior assumptions on the click model and the query specific parameters were able to capture the different characteristics of each query, for example, the clicking behaviour for navigational queries is different to that of informational queries [47], although this simple technique does run the risk of overfitting. A more sophisticated and accurate method such as probit Bayesian inference could be used to find optimal parameter settings [162], although for the purpose of this experiment this was not necessary.

The results of the experiments are summarised in Table (5.2). The Yandex column in the table refers to the ranking found in the search log itself. Despite the limitations already discussed, some notable observations can be made: 1) there are

significant differences between the click models, and in particular, the more complex dependent click model performed consistently well, 2) the training period significantly improved performance, demonstrating that the algorithm could be used to learn from existing search log data before being deployed in a practical setting, and 3) as expected, the myopic setting gave better results than the explorative setting owing to the limitations of the dataset. Ultimately, this experiment showed promising but inconclusive results, but serves to demonstrate the difficulty in evaluating an online algorithm that is responsive to user behaviour, which is discussed further in Chapter 7.

5.4.3 PAB Evaluation

A different approach was taken in the evaluation of the PAB algorithm. Again, a simulation was used, although in this case to test whether a diverse ranking of documents could be learned over time. The PAB approach was directly compared against the state of the art UCB1-Ranked Bandits Variant (UCB1-RBV) [23], which also used a mulit-armed bandit algorithm to learn diverse rankings of documents.

As in Section 5.4.1.1, M documents were chosen by the PAB algorithm over T stages, with a stochastic user 'clicking' on relevant documents at each time step t. Following the evaluation procedure defined for the UCB1-RBV algorithm, 20 simulated users were randomly assigned a subtopic preference in advance of evaluation. These were chosen using a Chinese restaurant process [163] (with $\gamma = 3$), resulting in an average of 6.5 unique subtopics for each query. Then, 50 documents were assigned to the subtopics in the same proportion that users were assigned to the subtopics.

At each iteration, $M = 5$ documents were selected by each algorithm and shown to a randomly chosen user. If the simulated user examined a document that had been assigned to their subtopic preference, a click was observed with probability p_{R}, otherwise with probability p_{NR}. These values were evaluated at each time step t up until a threshold of $T = 100,000$ and the results plotted in Fig. (5.2) and Fig. (5.3). In these plots, the performance is bounded above by the theoretical limit OPT, representing the optimal ranking, and below by $(1 - \frac{1}{e})OPT$, which is

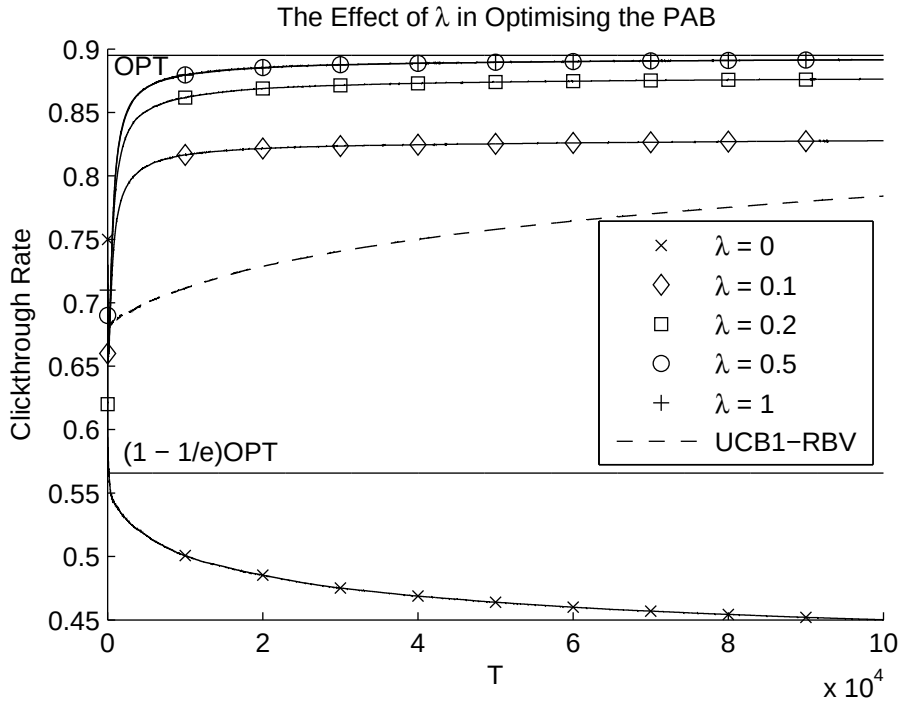


Figure 5.2: The effect that λ has on the clickthrough rate of the PAB and UCB1-RBV algorithms.

the worst case theoretical bound for the UCB1-RBV algorithm.

In Fig. (5.2), the impact of the diversification parameter λ was examined. Here, the *clickthrough rate* was measured, which was the proportion of rankings where at least one document was clicked. It can be seen that when $\lambda = 0$ the performance is very poor, performing worse than even the theoretical lower bound of the UCB1-RBV algorithm. This finding is the opposite of that found for UCB-DR, where performance was strong in the myopic, non-explorative case. Here, this indicates that the diversification of the results is responsible for the performance of the algorithm. Indeed, as the algorithm incorporates more diversification, its performance improves. This is because increased diversification increases the opportunity for user feedback on relevant subtopics, allowing the algorithm to learn an optimal ranking quicker. The PAB algorithm can also be seen to outperform the UCB1-RBV algorithm for all non-zero settings of λ.

In Fig. (5.3), the algorithm was tested to determine if it was resistant to random click noise. Because the PAB and UCB1-RBV algorithms do not incorporate click models like the UCB-DR algorithm does, this test was performed to ensure

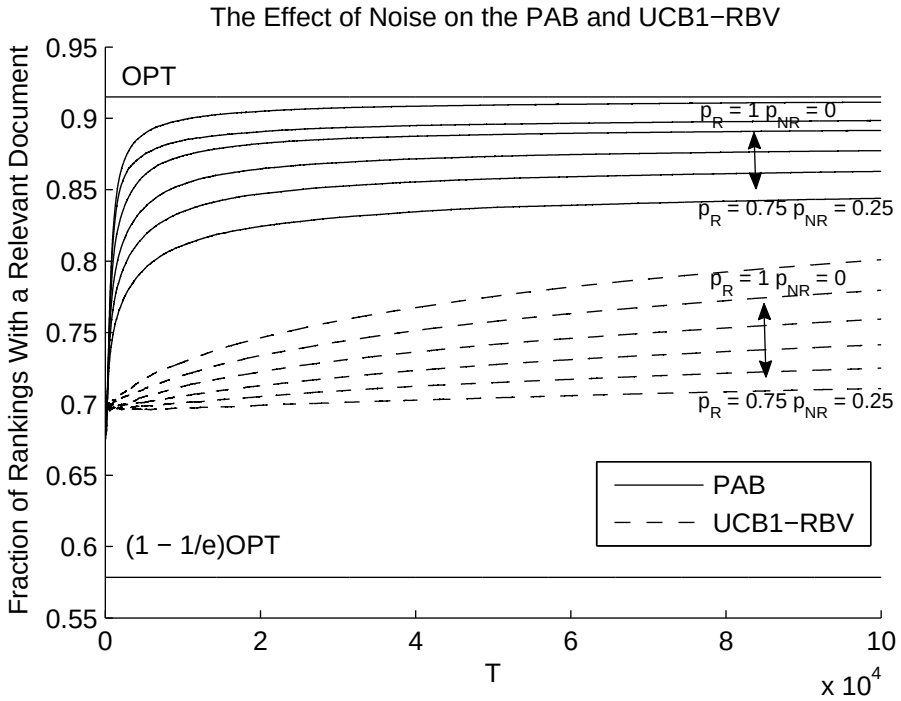


Figure 5.3: The p_{R} is decreased to measure how noise affects the performance (the fraction of rankings which contain a relevant document) of the PAB (where $\lambda = 1$) and UCB1-RBV algorithms over time.

that its rankings were still resistant to rank and other biases in click observations. Here, the values for p_{R} and p_{NR} were systematically adjusted to allow clicks to occur on non-relevant subtopics. Because the clickthrough rate was no longer a reliable measure of performance, the fraction of rankings with a relevant document was measured instead, which is equal to the clickthrough rate when $p_{\mathrm{R}} = 1$. The results in Fig. (5.3) show that as the click observations become less reliable, the performance of the PAB and UCB1-RBV algorithms degrades gracefully, and that the PAB approach still remains the more effective algorithm.

5.5 Conclusion

In this chapter, a model for dynamic online learning to rank has been presented, where the relevance of documents is learned over a period of time through the interaction with users, by considering clicks as observations of relevance. Through the theoretical derivation and analysis based on the *DIR* framework, it is shown that the relevance of documents can be updated over time using the rank-biased user clicks.

From this, the UCB-DR algorithm is defined based on the multi-armed bandit machine theory, and three click model based variants evaluated in simulations and over search log data. Aside from demonstrating the effectiveness of the algorithms under different problem settings, the experiments also highlighted the difficulty in evaluating *DIR* solutions and the steps that can be taken to combat these limitations.

An alternative formulation is also presented, which instead diversifies rankings of documents over time based on the portfolio theory of IR. In a different simulation, the resultant PAB algorithm was shown to outperform a similar state of the art approach, including when click noise was taken into account. This was a result of using only a single multi-armed bandit coupled with a proven diversification procedure, rather than inefficiently training multiple bandits simultaneously.

Chapter 6

Dynamics in Session Search Logs

Key to any research involving session search is the understanding of how a user's queries dynamically evolve throughout the session. When a user creates a query reformulation, they are consciously retaining terms from their original query, removing others and adding new terms. By measuring the similarity between queries, inferences can be made about the user's information need and how successful their new query is likely to be. By identifying the origins of added terms, the user's motivations can be inferred and an understanding gained of their interactions.

In this chapter, the *DIR* framework is used to define a novel term-based methodology for understanding and interpreting query reformulation actions. TREC Session Track data is used to demonstrate how the technique is able to learn from query logs and click data to test user interaction behaviour when reformulating queries. A range of term-based query reformulation dynamics are identified and evaluated that provide valuable insight into understanding query reformulation in session search.

6.1 Introduction

Session search is a dynamic information retrieval task that occurs when a user issues multiple queries consecutively to a search engine in the pursuit of satisfying one or more information needs. A session is typically defined as a period of continuous interaction with a search engine and can be demarcated in a number of ways, for example, by 30 minutes of inactivity [164]. Sessions containing more than one

query make up a significant proportion of search activity, with one study finding 32% of sessions contained 3 or more queries [27]. Understanding the underlying interactions in session search can lead to improved search interfaces, better search rankings and user satisfaction.

Sessions are driven by query reformulations, the user controlled act of modifying an existing query in order to pursue new search results. Query reformulations are usually closely related to the user's previous query and reflect the shifting cognition of the user throughout the session search. For instance, a user may have an unclear information need at the start of a session which becomes more refined as snippets are read and documents are clicked. Such queries can be ambiguous when the user is unsure how to explicitly define their information need [29] or explorative when the user is actively seeking a broad range of information on a subject [30]. In both cases, the information need can change throughout the session, whether through specialization or generalization, which leads to variations in the queries that are used.

Sessions are typified by queries consisting of core terms related to the underlying information need and additional terms that reflect the user's cognitive changes [165]. Over the course of the session, the core terms may change as well. At any point in a session, three possible *dynamic term actions* are available to a user:

Term Retention - Keeping terms from one query to the next, the core terms for the current information need.

Term Removal - Removing a term from a query.

Term Addition - Adding a new term not present in the preceding query to the query reformulation.

To illustrate a particular instance of query reformulation within session search and the described term actions, Table (6.1) contains the queries in a typical search session found in the 2013 Session Track dataset [1]. This session represents an

Table 6.1: Queries in session 40 of the TREC 2013 Session Track.

Impression Position	Query
1	`gun control opinions`
2	`gun control us government`
3	`gun control current affairs`
4	`gun control current affairs`
5	`gun violence us`
6	`law center to prevent gun violence`

explorative information need regarding public and political opinion on US gun control laws. The terms '`gun control`' are **retained** through the first four queries, with the user **adding** and **removing** terms '`opinions`', '`US government`' and '`current affairs`' in order to learn more about the topic. The focus shifts in query 5 with '`gun control`' changing to '`gun violence`', indicating a change in information need, which is expanded upon in the final query, which is more specialized.

Without knowing the underlying information need driving the queries, the example demonstrates that it is possible to infer persistent subtopics and the terms that are likely to be *retained* or *removed* from query to query (in this case '`gun control`' and '`gun violence`'). A certain degree of overlap is typical between queries but how much? What factors influence whether a term is likely to be kept or *removed* in the next query? Can one determine a source for the new terms that are introduced into a query? Measuring the similarity between queries and other sources of text can help to resolve some of these questions and to allow for the building of descriptive and evaluative models of dynamic user behaviour during a session search.

For instance, in the example session it can be observed that the snippets of all the results for the first query contain the terms '`gun control`', and out of all ranked documents only the clicked document (ClueWeb ID *clueweb12-0100wb-86-17546*) contains the terms '`US government`' (in the phrase "*US Government Info Guide*"), which were then used in the next query. One inference that could be drawn here is that the user observed the terms '`US government`' in the clicked

document, and this influenced their reformulation decision making process.

In this chapter, an understanding of the query reformulation process is gained and helps to resolve the following research questions:

1. What is the relationship between terms found in adjacent queries in search sessions? How often are terms from a query *retained* or *removed* in a query reformulation?

2. Where are query reformulation terms not present in the original query sourced from, and can term *addition* be modeled?

3. Can dynamic user-behaviour scenarios defined on terms that are *retained*, *removed* or *added* be informative of the quality of query reformulations?

These questions are resolved by introducing a novel methodology for interpreting query reformulations using terms. The technique is used to explore term *retention* and *removal* by analyzing adjacent and non-adjacent queries in sessions. With term *addition*, experimental observations indicate that a significant number of *added* terms in a reformulation can be sourced from the terms that the user was exposed to in the previous impression. An impression consists of a query, its snippets and its documents, all of which contain terms that the user may have encountered during session search. By also incorporating click information, three sources for such terms can be defined and evaluated: *clicked* and *non-clicked snippets* and *clicked documents*.

The next stage in this analysis involved measuring the value of the three term sources in determining whether query terms were *retained*, *removed* or *added*, leading to eight possible scenarios of user behaviour. To evaluate the effectiveness of scenario-based term prediction, and also the user's observed query reformulations, the experiments confirm whether the term actions ultimately lead to increased user satisfaction or improved search rankings. This is measured using implicit click information and a number of IR metrics.

The analysis was conducted on the TREC Session Track data from 2011 to 2014 [166, 167, 1], a set of standardized query logs comprising queries grouped by

sessions across a number of predefined topics, the ranked documents, their snippets and clickthroughs (including order and dwell time) and relevance judgements. The documents belong to the ClueWeb09[1] and ClueWeb12[2] corpora. This dataset was chosen as it is widely available, well regarded in the IR community and whilst small when compared to commercial query logs, it is rich with potential sources for term discovery (snippets and documents), interaction data (clicks and dwell time) and relevance judgements (for evaluation).

Unlike the earlier chapters in this thesis, this chapter deals with applying the *DIR* framework to a search log so that the inherent dynamic behaviour of the user within the dataset can be discovered. The dynamics in this case are those of the user rather than the search system, and so must be inferred rather than modeled. Here, no algorithm is defined and evaluated on test sets. Instead, the components of the dynamic framework are used as a motivation for defining the methodology in this chapter, for instance, the three term actions represent explicit dynamic actions available to the user during the act of query reformulation.

6.2 Related Work

This is not the first query log analysis of query reformulation behaviour. Jansen et al. [168] defined different query reformulation states and the transition patterns that occurred during a session and evaluated them over a large query log. Their research idea is similar to the scenario-based approach in this chapter, although in their study the states operate on a query level by looking at the degree of overlap between queries, rather than the term based approach used here, but some of the findings are similar. Lie et al. [169] explored a similar state-based analysis but this time on a user study that allowed them to determine different types of behaviour based on the type of task being performed by the user. Kinley et al. [165] also performed a user study with the intention of observing different query modifying behaviour (such as replacing, adding terms etc.) and linking it to a user's 'cognitive style' of query reformulation. A similar work to this is Huang and Efthimiadis' [170]

[1] `http://www.lemurproject.org/clueweb09/index.php`
[2] `http://www.lemurproject.org/clueweb12.php/`

classification of different types of reformulation behaviour which utilised clicks from query logs and used term differences as well. Nonetheless, this is the first such study using a purely term-based approach that also incorporates clicks in a user interaction model. The $\text{snip}_{t\text{LC}}$ definition and experiments in Section 6.5.1 are based on the examination hypothesis model [38] which is based on eye-tracking research. Another recent eye tracking study [171] found that when browsing search results, users will glance at snippets but not fully read them, returning to them at a later point if at all.

The work by Guan et al. [34] on session search re-ranking based on query and impression term matching is a similar approach to this, although here a more complex model is built that captures the dynamic user interactions, and also document re-ranking is not the objective. Another similar work is by Jiang et al. [55] who conducted a comprehensive user and eye tracking study to understand how users behave over the course of a session. Their work included statistics on reformulation behaviour and ranking metrics across queries in sessions and many of their results mirror those in this chapter. Both of the described pieces of research can be seen as a specialization of the methodology in this chapter (for instance focusing on a particular type of term source) that concerns a specific IR problem, whereas the work in this chapter is a more general study on trying to understand dynamic reformulation behaviour.

The work most similar to this is by Liu et al. [172] who used terms from clicked snippets to aid in query recommendation. They also recognized that information needs persist through adjacent queries in search sessions but are difficult to define based purely on previous queries, and so use snippets as an additional term source. Unlike the methodology in this chapter, they only use clicked snippets whereas here terms from non-clicked snippets and documents are also incorporated, as well as those from the previous query. Where the work mainly diverges is that their objective is to locate terms that are useful for query recommendation, whereas the objective here is to identify useful term sources for query reformulation (of which clicked snippets is one) under a number of conditions including clicks, rank and

Table 6.2: TREC 2011, 2012, 2013 and 2014 Session Track data overview.

	TREC Session Track			
	2011	2012	2013	2014
Number of topics	62	48	49	51
Number of sessions	76	98	116	1075
Number of impressions	280	297	471	3784
Number of $q_t \rightarrow q_{t+1}$ pairs	204	199	355	2709
Average number of terms in query	3.34	3.40	3.51	3.21

impression position.

The work in this chapter differs from the literature in that: 1) the methodology is term-based rather than query or task-based, 2) the methodology is derived from data rather than a user or eye-tracking study, and 3) the model is based on the *DIR* framework, incorporates clicks and differentiates term sources such as snippets and documents as sources of reformulation terms.

6.3 Analytical Setup

Experiments were conducted using the TREC 2011, 2012, 2013 and 2014 Session Track data [167, 1], which contained search logs collected by the TREC organizers and grouped by session. While particpants were given predefined topics to search over, the organizers recorded all of the displayed URLs, titles and snippets and also user interactions including clicks and document dwell time . The corpora used were the ClueWeb09[1] and ClueWeb12[2] datasets. Relevance judgements were also collected for documents related to each of the topics. See Table (6.2) for more detailed information about the datasets.

In comparison to commercial search logs, the TREC dataset is small. Moreover, the artificial setting in which the participants were recorded conducting session search makes analysis on its data difficult to apply to commercially used search systems. For the purpose of this study, the dataset is ideal in that it was the only publicly available search log that contained the rich impression data needed for this analysis, that is, clicks, dwell times and all ranked snippets and documents (not just clicked). While the statistics in this chapter may not exactly reflect those found in commercial logs, the theoretical insights are still transferable, can be readily

reproduced, and the methodology is applicable to any similarly rich dataset. Furthermore, the dataset proved large enough to give statistically significant values in the experiments.

Sessions in the dataset are made up of a list of queries, each of which contains a ranking of M documents (typically $M = 10$), the snippets and titles of each document and a list of the documents that were clicked, including their order and dwell time. In a session containing T queries, the t'th query is referred to as q_t and its query reformulation (if $t < T$) as q_{t+1}. In this chapter, the boldface variable $\vec{\mathbf{q}}_t$ denotes the *term vector* representation of the query (with term frequency as the term weights) and the capitalised Q_t as the *set* of its terms d_t. This chapter's analysis and experiments concern the changes between queries in a session, so each pair of queries $q_t \rightarrow q_{t+1}$ is extracted for $t = 1, \ldots, T-1$.

An **impression** refers to all of the search data related to a query such as the ranked list of documents and the clickthroughs. Elements of an impression include snippets (and their titles), clicks, dwell time and documents. In this dataset each session ends with a 'test' query intentionally containing no ranking, the original purpose being for researchers to create rankings for this query by utilizing the information in the session. In these cases the query is not considered to have an impression but it is still made use of in the query reformulation pairs unless stated otherwise.

The Natural Language Toolkit (NLTK)[3] was used to remove punctuation and stop words and tokenize all textual content, and then the terms were stemmed using the Porter Stemmer [173]. Consideration was given to the choice to remove stop words, as it did render some query reformulations as identical to the previous query, even if they originally weren't. For instance, in session 95 of the 2012 dataset, $q_1 =$'`connecticut fire academy`' and $q_2 =$'`what is the connecticut fire academy`', yet after stop word removal $q_1 = q_2$. In this case, the reformulation is a more focused query than its predecessor but it nonetheless addresses the same information need with the same core terms. The Beautiful

[3] `http://www.nltk.org/`

Soup HTML Parser[4] was used to extract textual content from the ClueWeb HTML documents.

Each term source (such as a query or snippet) was treated as a bag of words (BoW), even though using n-grams could make the methodology more discernible. For example, in session 285 of the 2014 dataset, q_1 =‘`depression`’ and q_2 =‘`help someone with depression`’. With BoW, the terms ‘`help`’ and ‘`someone`’ are treated separately, and there are indeed examples of the term ‘`help`’ in the snippets for q_1, although erroneously in the context of the web-page (‘`...Help FAQ Advertising...`’ at rank 3) rather than that implied by the query. Here, a bigram would distinguish ‘`help someone`’ in the correct context. Nonetheless, all of the similarity measures used operate on a BoW model, and given that typically only 1 or 2 terms are observed being *added* or *removed* from adjacent queries in a session, a unigram model is sufficient in this case.

This methodology concerns the analysis of text similarities. The similarities of queries are measured using the following formulae:

$$Jaccard(Q_1, Q_2) = \frac{|Q_1 \cap Q_2|}{|Q_1 \cup Q_2|} \tag{6.1}$$

$$Cosine(\vec{\mathbf{q_1}}, \vec{\mathbf{q_2}}) = \frac{\vec{\mathbf{q_1}} \cdot \vec{\mathbf{q_2}}}{\|\vec{\mathbf{q_1}}\| \cdot \|\vec{\mathbf{q_2}}\|} \tag{6.2}$$

where q_1 and q_2 are queries (or any other term source). *Jaccard* similarity is commonly used in measuring set similarity, in this case sets of terms, and *Cosine* similarity is widely used in the vector space model in IR.

6.4 Term Retention and Removal

In this first analysis, the term actions *retention* and *removal* are investigated. These two actions are only applied to terms d_t found in the user's query Q_t, where *retention* means that $d_t \in Q_{t+1}$ and *removal* is when $d_t \notin Q_{t+1}$.

The average number of terms *retained*, *removed* or *added* were measured as was the average Jaccard and Cosine similarity between adjacent queries found in

[4] `http://www.crummy.com/software/BeautifulSoup/`

Table 6.3: Average number of terms *retained*, *removed* or *added* from $q_t \rightarrow q_{t+1}$ and the similarity between the two queries across TREC Session Track datasets.

	TREC Session Track				
	2011	2012	2013	2014	Combined
$Jaccard(\vec{\mathbf{q}}_t, \vec{\mathbf{q}}_{t+1})$	0.49	0.52	0.50	0.51	0.50
$Cosine(Q_t, Q_{t+1})$	0.60	0.65	0.62	0.63	0.63
# terms *retained* from $q_t \rightarrow q_{t+1}$	2.12	2.29	2.28	2.10	2.13
# terms *removed* from $q_t \rightarrow q_{t+1}$	1.20	1.05	1.20	1.11	1.12
# terms *added* from $q_t \rightarrow q_{t+1}$	1.33	1.35	1.33	1.21	1.24

sessions in the TREC datasets, the results are in Table (6.3). It can be seen that adjacent queries are similar to one another, with high similarity scores and term *retention*. The measures are generally consistent across the individual datasets and their combination, and so the remainder of these analyses will be conducted on the combined dataset. Across all datasets, an average of 63% of the terms in q_{t+1} can be found in q_t, where 66% of its terms are *retained* (2.13 terms), 34% of terms are *removed* (1.12 terms) and 1.24 terms are *added*. 33% of the time the reformulation contains all of the terms found in the original query. *Retained* terms clearly make up a large proportion of a reformulation and are indicative of the core terms defining the user's information need.

An important observation is that on average the length of queries increases from 3.25 terms to 3.37 terms, meaning that it cannot always be possible to source q_{t+1} terms from q_t. To determine if this relationship holds throughout a session, the average query length at each impression position was found for a number of different session lengths (see Fig. (6.1)). The results show that for shorter sessions (2 - 4 impressions) query size does appear to marginally increase, for medium session lengths (5 - 7 impressions) the query size initially increases to a point and can start to decrease, and for longer sessions (8 - 10 impressions) the query length varies unpredictably, presumably due to the small population sizes. Medium and longer sessions are likely to contain shifts in information need (for example, between queries 4 and 5 in Table (6.1)), which may explain the variability of query length with increased impression position. It is clear from these results that reformulations can gain or

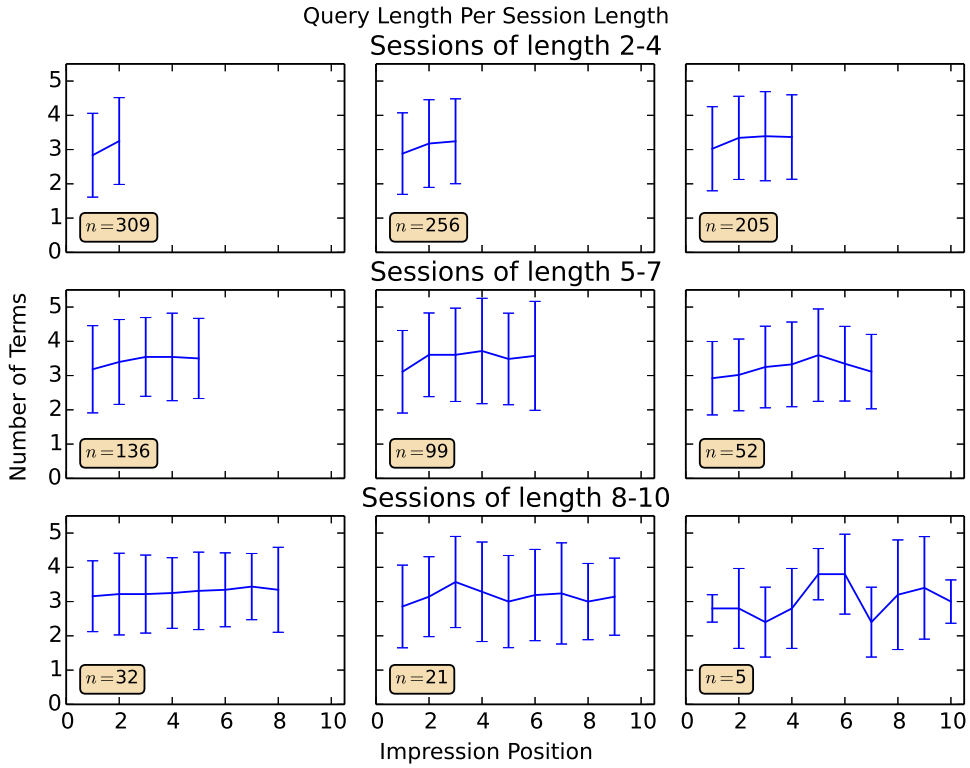


Figure 6.1: Plots of the average number of terms in queries at different impression positions in a session, for different lengths of session. The number of instances of each session length are labelled as n in each subplot.

lose terms depending on their position in a session.

In Fig. (6.2), the similarity between query reformulations and their preceding query was measured at each impression position. In the previous analysis it was found that impression position affected query length (subject to session length), so here it is investigated whether this also holds for query similarity. The main conclusion that can be drawn is that the results are too variable to discern a pattern, with no clear trend for increasing or decreasing similarity. What this indicates is that throughout a session, queries are generally similar to their reformulations regardless of their position in the session.

Nonetheless, one would expect information needs to change throughout a session and when that happens the similarity between adjacent queries should change. For instance, in Table (6.1) the average similarity scores between all adjacent queries are $Jaccard = 0.44$ and $Cosine = 0.57$, but between queries 4 and 5, the

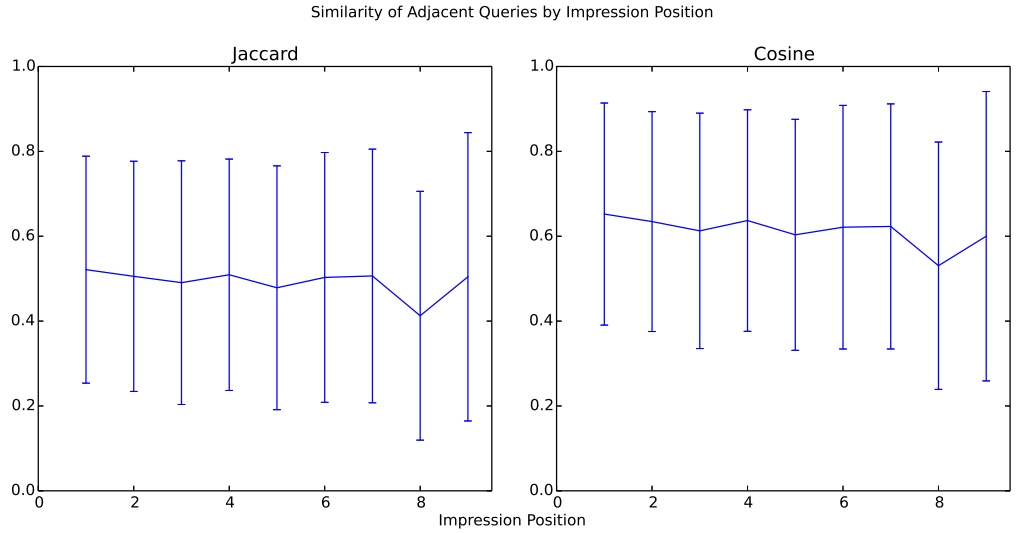


Figure 6.2: Average similarity of $q_t \rightarrow q_{t+1}$ pairs for impression positions $t = 1 \ldots 9$.

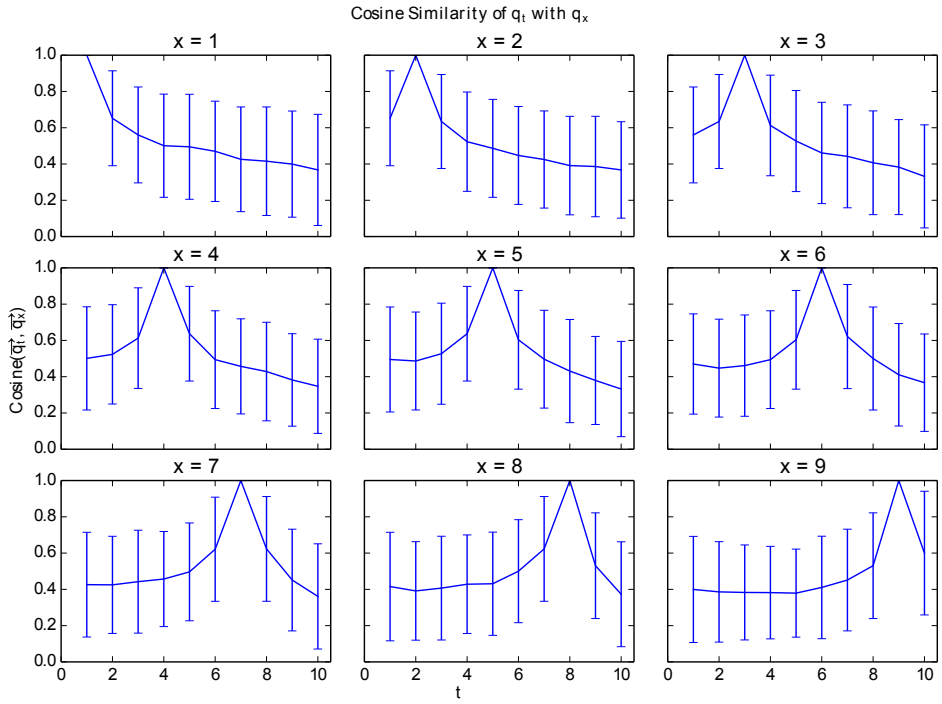


Figure 6.3: Cosine similarity of fixed query q_x with every other query q_t in the session for $x = 1 \ldots 9$.

shift in query intent is captured in the change in similarity scores, calculated as $Jaccard = 0.17$ and $Cosine = 0.29$, a noticeable departure from the average.

In Fig. (6.3) it can be seen that the core query terms do not remain constant throughout a session, indicating that the terms used in queries are always progres-

sively changing. In this instance, the cosine similarity measure was measured although the same trend can be observed for Jaccard similarity. It can be seen that queries occurring on either side of the 'fixed' query q_x are the most similar but queries further away in the session become more dissimilar. This behaviour holds regardless of the position of q_x in the session. Together with the previous result, this demonstrates one of the key motivations of this methodology, that there does not exist a set of 'core' terms that represent the user's information need throughout the session, instead, the query and its core terms dynamically evolve as the user's information need changes. Queries at the start of a session can be very different from those at the end, and as such, term *retention* and *removal* are useful locally with adjacent queries but less so across the whole session.

6.5 Term Addition

So far it has been shown that on average 63% of the terms in query reformulations can be explained by the *retained* or *removed* term actions, leaving 37% of terms unaccounted for. In this section the *addition* term action is investigated, which is applicable to *added* terms add_{t+1} which are terms added from q_t to q_{t+1} i.e. the set of terms $\text{ADD}_{t+1} = Q_{t+1} \backslash Q_t$. Whereas before the analysis concerned the similarity of the query reformulation against query terms d_t, in this section the similarity is measured against terms from each of the term sources found in the impression.

When different term sources are compared with add_{t+1} there are some problems caused by term source length. For instance, the *Jaccard* similarity is sensitive to the size of the sets it compares; comparing with a larger set leads to lower similarity, making comparisons between different term sources biased. Additionally, in the studies so far, the comparisons have been over the small number of terms found in queries, where every term can be considered important. On the other hand, the term sources can contain hundreds of terms, only a few of which may match the *added* terms.

These problems are counteracted in two ways: first, the TFIDF scores [44] are used instead of term frequencies in the *Cosine* similarity measure, which helps to

match on those *added* terms that are important to the term source. Thus, from this point on any term vectors $\vec{\mathbf{q}}_t$ refer instead to the TFIDF vector. Secondly, the BM25 score [2] (with typical parameter settings $k_1 = 1.2$ and $b = 0.75$) is measured. This score was designed to find the similarity of queries consisting of a few terms against documents with many terms, and so is robust to variable document lengths. For both of these measures, the IDF must be calculated over the document collection, as well as the average document length for BM25. In this methodology, the collection of all instances of a term source are treated as the document collection when analysing terms in that term source. For example, when analysing the snippet term source, the collection of all of the snippets in the dataset are considered as the data collection for calculating IDF and the average document length.

6.5.1 Snippet Analysis

The analysis here starts by considering the snippets found in an impression. A query q_t may have up to M ranked snippets snip_{ti} where snip is the snippet and i is its rank position $1 \leq i \leq M$. In the datasets the snippet title is joined onto the snippet under the assumption that anyone reading the snippet has also read its title.

In the first study the similarity of snippets snip_t against *added* terms add_{t+1} are investigated at different rank positions. A natural hypothesis based on eye tracking studies [174] is the concept of rank bias, that search results ranked at the top have a higher chance of being observed, thus, they should be more similar to terms *added* to the next query than lower ranked, potentially unobserved snippets.

In Table (6.4) the similarity scores for each snippet snip_{ti} from rank 1 to rank i in the impression are averaged. Under the assumption given by the Examination Hypothesis model that users examine all snippets in order from the top of the search results to the bottom, *all* snippets up until rank i are averaged over, not just the snippet at that rank. The results show that across metrics the similarity peaks at rank positions 2 and 3 before dropping with each rank. The similar lengths of snippets at each rank means that a term source length bias can be ruled out. Curiously, the highest ranked snippet on its own does not have the highest similarity to *added* terms. The implication here is that terms used in query reformulations have a higher

Table 6.4: Average similarity scores between *added* terms add_{t+1} and snippets snip_t up to rank i in an impression. For example, if $i = 3$, then the score is the average over $\text{snip}_{t1}, \text{snip}_{t2}$ and snip_{t3}. Maximum values for each similarity measure are in boldface.

	i				
	1	2	3	4	5
$Jaccard(\text{ADD}_{t+1}, \text{SNIP}_{ti})$	0.00531	**0.00536**	0.00529	0.00507	0.00494
$Cosine(\overrightarrow{\text{add}_{t+1}}, \overrightarrow{\text{snip}_{ti}})$	0.0184	**0.0197**	0.0195	0.0187	0.0185
$BM25(\text{add}_{t+1}, \text{snip}_{ti})$	0.704	0.756	**0.758**	0.737	0.728
# terms in snip_{ti}	48.3	48.8	49.9	50.2	50.3

chance of being found in the top 2 or 3 ranked snippets and that users do not just consider the top ranked snippet on its own. This examination of the top 2 or 3 search results is consistent with eye tracking observations.

From click model research, the assumption can also be made that if a click is observed in an impression, then the user has examined all snippets up until the rank position of that click. Let LC be denoted as the rank of the Last Click in an impression (that is, the lowest ranked clicked document). In this next study, the aim was to observe whether similarity change occurs at rank LC and for the snippets ranked above and below it, akin to the 'Click > No-Click Next' strategy and its variants outlined by Joachims et al. [53]. If an impression didn't contain a click, then all snippets in the impression are included, the results are in Table (6.5).

It was expected that a decrease in similarity following the rank LC would be observed, owing to the hypothesis that a user does not examine documents ranked lower than the last click. In this experiment it can be seen that this is not the case, a higher similarity score is observed when considering all snippets in an impression rather than just up until the last clicked. A difference between this session search setting and that typically modeled with click models is that in this case, even after a document has been clicked, it is known that the user returned to the set of search results in order to issue a reformulation. Conventional click models do not take into account multiple queries in a search session. As such, in this case it is likely that the user continued to examine snippets after the last click, before abandoning the query and issuing a reformulation, leading to the observed results. Also, by comparing

Table 6.5: Average similarity scores between *added* terms add_{t+1} and snippets snip_t up to and around rank LC in an impression, as well as all snippets. Maximum values for each similarity measure are in boldface.

	i				
	LC − 1	LC	LC + 1	LC + 2	M
$Jaccard(\text{ADD}_{t+1}, \text{SNIP}_{ti})$	0.00440	0.00446	0.00450	0.00458	**0.00465**
$Cosine(\overrightarrow{\text{add}_{t+1}}, \overrightarrow{\text{snip}_{ti}})$	0.0167	0.0171	0.0172	0.0174	**0.0175**
$BM25(\text{add}_{t+1}, \text{snip}_{ti})$	0.656	0.671	0.676	0.682	**0.688**
# terms in snip_{ti}	51.0	51.0	51.0	50.9	50.4

these results with those in Table (6.4) it can be seen that the top ranked 2-3 snippets are still more likely to contain *added* terms.

These inferences can be observed in the example session in Table (6.1). For queries q_5 =‘`gun violence us`’ and q_6 =‘`law center to prevent gun violence`’, where q_6 has already been noted for its shift in query intent, the impression for query q_5 has the *added* term ‘`center`’ in the snippet at rank 3, which is the last (and only) clickthrough. This is in line with the findings on top ranked snippets in Table (6.4) and corroborates the last click hypothesis above. Yet, at ranks 7 and 8 there are instances of the *added* term ‘`prevent`’, suggesting that in this case the user examined snippets beyond the one that was clicked.

6.5.2 Term Sources

So far the effect of impression and rank position on similarity has been introduced and in the last experiment clicks were introduced. Here, clicks are directly used to further distinguish between the two distinct sources of *added* terms in an impression: snippets and documents. This means that an impression can be split into three term sources:

Non-Clicked Snippets (ncs) Snippets without a clickthrough.

Clicked Snippets (cs) Snippets with a clickthrough.

Clicked Documents (cd) Documents with a clickthrough

Note that the combination of nc and cs gives all snippets in the impression i.e. $(\bigcup \text{CS}) \cup (\bigcup \text{NCS}) = \text{SNIP}_M$. From this, impression terms can now be considered

Table 6.6: Average similarity of *added* terms with click-based variations of the snippet and document term sources and also the full preceding impression (imp) and all previous impressions (hist). Bold scores indicate a statistically significant ($p < 0.01$ under Welch's t-test) difference from non-clicked and 'All' variants of the term source.

Term Source	# terms	*Jaccard*	*Cosine*	*BM*25
All Snippets (snip_M)	50.4	0.00465	0.0175	0.688
Clicked Snippets (cs)	50.1	**0.00752**	**0.0289**	**1.100**
Non-Clicked Snippets (ncs)	50.5	0.00445	0.0167	0.660
All Documents (ad)	808.8	0.00131	0.0251	5.612
Clicked Documents (cd)	974.2	**0.00171**	**0.0398**	**8.207**
Non-Clicked Documents (ncd)	796.4	0.00128	0.0240	5.417
Impression (imp)	8127.2	0.00067	0.0381	3.535
Historical (hist)	19802.9	0.00052	0.0568	4.370

as belonging to one or more of the described term sources and so an evaluation was conducted on how effective they are at providing *added* terms for query reformulations. The reasoning for incorporating clicks into the term source definitions is that implicit user feedback is an indicator of the relevance of the terms contained in the source and the user's behaviour at that point in the session.

Table (6.6) contains the results of the similarity analysis over different term sources and their variations with *added* terms. Clicked snippets and documents (cs and cd) were compared with their non-clicked counterparts (ncs and ncd) and also against both combined (snip_M and ad). In both cases, statistically significant increases in similarity are observed when considering clicks, a clear indicator that clicked snippets and documents are a source of terms used in query reformulations. Clicked documents score higher than other term sources for the length normalised metrics *Cosine* and *BM*25 (the score is lower for the length biased *Jaccard* measure), indicating the importance of clicked documents. The similarity of non-clicked documents was measured in order to provide comparison with clicked documents, but ultimately in this chapter they're not considered as a term source. This is because it cannot be known if the user has been exposed to them during the session, although it is feasible that the user has encountered the document before, or was satisfied by the non-clicked snippet itself.

The similarity with all of the terms found in the impression was also measured, where IMP = $\text{SNIP}_M \cup \text{CD}$ (not including the query). It can be seen that differentiating an impression into click-based term sources does lead to improved similarity scores than combining them all together into one impression-based term source. Taking this further, historical impressions were also measured, i.e. all impression terms that occurred earlier in the session up to and including impression t, given as $\text{HIST}_t = \bigcup_{k=1}^{t} \text{IMP}_k$, to test the assumption that users obtain terms not just from the preceding impression but also those encountered earlier. For instance, in the example in Table (6.1), the term '`current`' from q_3 is not found in the preceding impression for q_2, whereas it occurs 3 times in the snippet at rank 3 of q_1. There is an increase in similarity scores over the historical impression terms and values that is comparable with the other term sources, suggesting that terms can be sourced from earlier in the session. In this chapter, the term sources are defined based only on the preceding impression, but using earlier impressions could prove an interesting extension.

6.5.3 Dwell Time

From Table (6.6) it can be seen that clicked documents have substantially more terms than snippets. A central argument of this methodology is that users choose reformulation terms that they have been exposed to from term sources, hence, in order to come across terms in a long document, time must be spent reading it. The dataset records the dwell time of each clicked document, which is an indicator of reading time.

The average dwell time is 35.3 seconds before users return to the set of search results. This is similar to the 30 second threshold used in other IR research as a marker for a satisfactorily (SAT) clicked document [164]. SAT clicks are often used as a replacement for relevance judgements in the absence of human assessors, usually on large query logs. With this dataset a dwell time threshold of 30 seconds differentiates 40% of the clicked documents.

To test whether dwell time should be considered a feature in this methodology, the similarity of clicked documents against *added* terms was measured at a range of

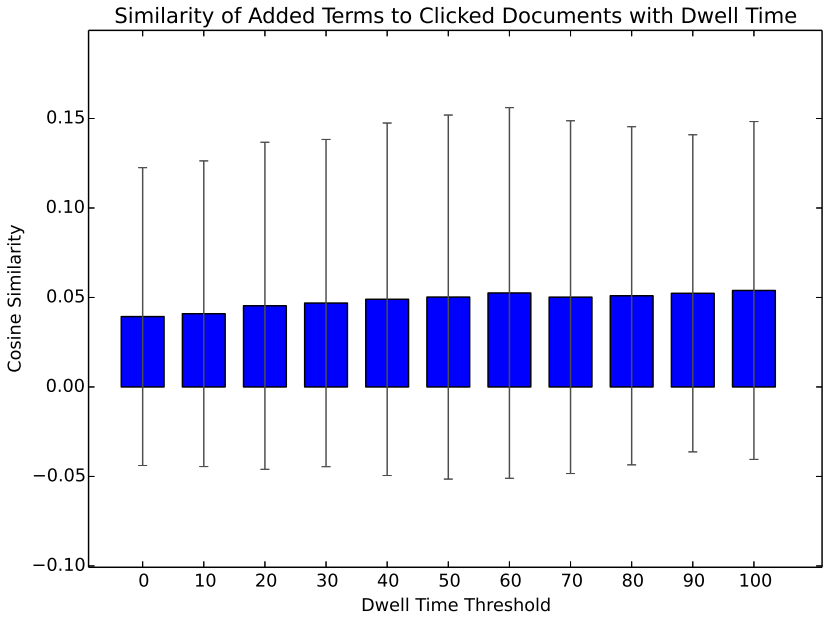


Figure 6.4: Average *Cosine* similarity of *added* terms with clicked documents at different dwell time threshold levels.

different dwell time thresholds. Fig. (6.4) displays the results for *Cosine* similarity, the other measures reported similar findings. Whilst one can observe a slight increase in similarity with dwell time threshold, the results are too variable to be able to draw any conclusions. In particular, the SAT click threshold does not appear to offer any clear indicator of improvement. These findings are supported by recent research that argues that this single value cannot capture the complexities of reading behaviour and user satisfaction [104]. As such, in this chapter dwell time is not considered as a feature in the methodology and instead all clicked documents are used as a term source collectively.

6.6 Term Scenario Analysis

The term-based methodology in this chapter has given insight into the circumstances where terms are *retained*, *removed* or *added* to query reformulations. Use of the similarity measures has helped to define the three term sources, based on user interactions, that influence the terms *added* to the next query in a session. In this section, the methodology is extended to measure how effective query reformulations are under different circumstances. This is achieved by defining 8 user

behaviour scenarios based on the combination of term sources, which can help to interpret the analysis results and to understand user dynamics.

6.6.1 Query and Added Term Scenarios

The first focus in this section is on the query terms d_t and whether the term actions *retention* or *removal* are usually applied to them by the user. To expand on the limited information available on the terms in the query, one can look for occurrences of the term in the impression. More specifically, the three term sources ncs, cs and cd. A query term d_t can belong to any combination of term sources, including all or none, giving 8 query **term scenarios**. Each combination of term source defines a scenario and a full index of scenario number definitions is given in Table (6.8).

In the previous analysis, inferences were made on terms based on which term source they originated from. With the expansion of 3 term sources to 8 scenarios, more interesting observations can be made. For instance, in the first query in the example in Table (6.1), the terms '`gun`' and '`control`' both belong to scenario 8 (they appear in non-clicked and clicked snippets and also clicked documents) and they are *retained* in the query reformulation. Conversely, the term '`opinions`' is only found in non-clicked snippets (scenario 5) and is subsequently *removed*. An inference that can be made here is that finding query terms in clicked snippets and documents is a strong indicator that the term will be kept, whereas query terms that only appear in non-clicked snippets are more likely to be *removed*.

Added terms are also assigned to the same scenarios in Table (6.8). Given that the purpose of this methodology is to understand when terms from the previous impression (including its query) will be used in the reformulation, it is appreciated that a real search system would not have access to *added* terms in order to assign them to scenarios. Nonetheless, by analyzing these terms in the same way as query terms, one can gain insight into which circumstances a user is likely to add terms from the impression.

All query reformulation pairs were extracted from the dataset as before but this time did not include test queries (the final query in each session). Test queries did not contain rankings or relevance judgements, and thus were unsuitable for the

Table 6.7: Overall average clicks, non-clicks and documents per impression and overall number of query and *added* term scenarios.

# ranked documents	10.5
# clicks	0.626
# non-clicks	9.87
# query term scenarios	7621
# added term scenarios	2981

Table 6.8: Scenario number definitions, occurrence % for query and *added* term scenarios and average number of ranked documents and clicks for each scenario.

Scenario	$d \in$			Query term scenarios			Added term scenarios		
	ncs	cs	cd	%	# Docs	# Clicks	%	# Docs	# Clicks
1	False	False	False	9.95	8.64	0.27	57.0	10.2	0.38
2	False	False	True	0.35	11.1	1.89	7.85	10.4	1.86
3	False	True	False	0.05	3.50	1.25	0.20	7.00	1.00
4	False	True	True	0.68	10.2	2.27	2.01	11.0	2.17
5	True	False	False	60.2	10.5	0.06	24.2	10.8	0.15
6	True	False	True	2.27	10.6	1.41	4.43	10.4	1.46
7	True	True	False	0.42	11.7	0.94	0.07	10.0	0.50
8	True	True	True	26.1	10.9	1.70	4.26	11.7	1.91

evaluations in the next subsections. Terms from q_t and add_{t+1} were assigned to each scenario and an overview of the results are in Tables (6.7) and (6.8).

In Table (6.8) it can be seen that both sets of term scenarios are variably distributed. Scenario 5, which refers to the case where terms only appear in non-clicked snippets, is the most common scenario for query terms, comprising 60.2% of the data. For this scenario the average number of clicks is 0.06, well below the overall average in Table (6.7). Thus, scenario 5 appears to be capturing the common case where users do not click on any results, hence no other term source matching occurs. Scenario 8 makes up a further 26.1% of cases and represents the situation where terms appear in all term sources. One would expect query terms to appear in snippets (either ncs or cs) and it can be seen that this is the case 90% of the time. Interestingly, 9.95% of query terms do not appear in the impression at all.

A different distribution of scenarios for *added* terms is observed, the most prominent being scenario 1 at 57%. This is the case where *added* terms cannot be found in the previous impression and mirrors the findings in Table (6.3). Scenario

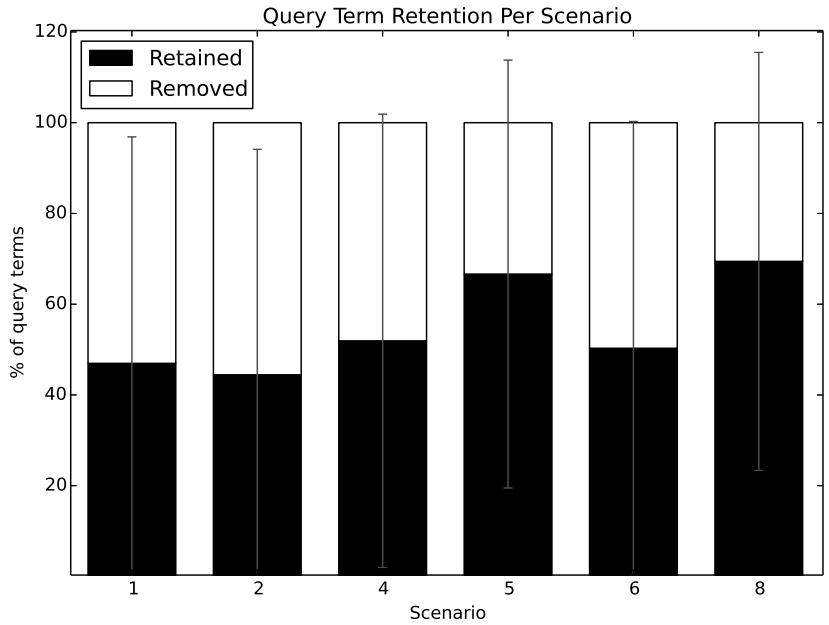


Figure 6.5: Proportion of query terms that are *retained* or *removed* per term scenario.

5 is also common for *added* terms. The four scenarios where terms are found in clicked documents (2, 4, 6 and 8) make up 18.6% of the scenarios, further evidence of clicked documents being a valuable source of *added* terms. There is a noticeable difference in occurrences between query and *added* terms in scenarios 2 and 4. Scenarios 3 and 7 rarely appear for both query and *added* terms; this can be explained by the fact that these are the cases where terms appear in clicked snippets but not clicked documents. Given that the snippet is derived from the document itself, this makes it unlikely for these scenarios to occur, and they are ignored in subsequent analyses.

6.6.2 Term Actions

Query term scenarios fall into two term action categories, *retained* or *removed*. Fig. (6.5) shows the proportion of query terms that are *retained* or *removed* from query reformulations for each scenario. The first observation here is that the two most common scenarios (5 and 8) lead to high term *retention* rates which are around the overall average term *retention* of 66%. This coincides with the earlier finding that users generally *retain* terms between adjacent queries, thus, the core terms are falling into these scenario numbers. For example, in Table (6.1) the query terms

'`gun control`' both belong to scenario 8 for the first 2 queries and are *retained*. For queries 3 and 4 (which are identical), they change to scenario 5 and are then subsequently *removed* in the next query.

Scenarios 2, 4 and 6, which capture instances of query terms appearing in clicked documents, occur infrequently for query terms and here seem to lead to the *removal* of terms. One inference is that terms appearing in clicked documents may be *removed* in lieu of the user satisfying that particular search intent. Low *retention* for query terms that are not found in the impression at all can also be observed, potentially an indication that the term was not useful in helping the user's search.

6.6.3 Term Scenario Evaluation

So far, some understanding of the term actions *retention*, *removal* and *addition* has been made without explicitly evaluating whether or not they are beneficial to the user. Simply understanding the dynamics of terms from queries and term sources, based on user behaviour in search logs, does not necessarily mean that they will improve the search experience. These evaluations demonstrate that this methodology is able to differentiate scenarios which may lead to future clicks or improvements in IR metric scores.

6.6.3.1 Click Based Evaluation

The first evaluation method involved observing whether the next impression in the session contained a click observation, an implicit measure of success and one tied to the user whose session was being analysed. In this experiment, for each term scenario the proportion of times each of the three term actions (*retaining*, *removing* or *adding*) led to a click in the next impression was measured and the results are in Table (6.9). Firstly, it can be seen that all term actions in scenarios 1 and 5 (where terms are not found in clicked snippets or documents) are less likely to lead to a click. When clicked documents are taken into account (scenarios 2, 4, 6 and 8) the likelihood of a click in the next query is much higher. In particular, for scenarios 2 and 4 clicks were more likely after removing query terms then *retaining* them, a result mirroring what was found in Fig. (6.5). Terms *added* from clicked documents

Table 6.9: Percentage of term scenarios and term actions that led to a click in the next query.

Scenario	% (Term action → Click)		
	Retained	Removed	Added
1	22.5	29.1	26.3
2	25.0	53.3	54.3
4	59.3	68.0	63.3
5	24.5	22.1	27.5
6	41.4	52.3	59.1
8	52.3	49.1	64.6

and snippets were also highly likely to result in a click.

6.6.3.2 IR Metric Based Evaluation

While clicks are important implicit signals of relevance, one can also make use of the TREC Session Track relevance judgements to evaluate the effectiveness of term actions. The majority of sessions in the dataset are linked to topics, for which documents have been assessed for relevance by human assessors on a scale from 0 to 4. For each impression in the dataset the Normalised Expected Reciprocal Rank (NERR) at rank position 10, nDCG at position 10 and the MAP were calculated. These metrics are widely used and well regarded in the IR community and the cutoff point at rank 10 was chosen in order to evaluate the quality of results in a typical impression. NERR is a metric that rewards displaying a highly relevant document at a high rank, nDCG measures the quality of the retrieved documents and their order and MAP balances precision and recall.

The difference in scores for each of the metrics calculated for the rankings of q_t and q_{t+1} were measured across each scenario and term action and the results are in Table (6.10). It can be seen that when scenario 1 query terms are *retained* there is a significant improvement across all IR metrics, but otherwise for the other scenarios the scores decrease, significantly so for scenario 8. A similar pattern across all scenarios can also be seen when terms are *removed*. Finally, for *added* terms the IR metrics decrease across all scenarios, significantly so for scenarios 1 and 2. These results indicate the existence of a general trend of decreasing IR score for adjacent queries, and it was found that a plot of the scores across impression

Table 6.10: Change in value for metrics nDCG, NERR and MAP from $q_t \rightarrow q_{t+1}$ for each term action and term scenario. Bold values indicate a statistically significant difference in IR metric score ($p < 0.05$ under the Wilcoxon signed rank test).

		Scenario					
		1	2	4	5	6	8
Retained	nDCG	**0.078▲**	−0.205▼	**-0.120▼**	−0.009▼	−0.019▼	**-0.058▼**
	NERR	**0.080▲**	−0.216▼	**-0.146▼**	−0.004▼	−0.024▼	**-0.064▼**
	MAP	**0.004▲**	**-0.011▼**	0.010▲	0.001▲	−0.003▼	**-0.009▼**
Removed	nDCG	**0.058▲**	0.000	−0.069▼	0.006▲	0.006▲	**-0.148▼**
	NERR	**0.037▲**	−0.024▼	−0.063▼	−0.015▼	0.017▲	**-0.140▼**
	MAP	**0.005▲**	−0.004▼	0.000	**-0.003▼**	0.001▲	**-0.010▼**
Added	nDCG	−0.025▼	**-0.127▼**	−0.051▼	−0.023▼	−0.046▼	−0.091▼
	NERR	**-0.019▼**	**-0.123▼**	−0.082▼	−0.020▼	−0.052▼	**-0.073▼**
	MAP	**-0.007▼**	**-0.007▼**	0.002▲	0.000	−0.006▼	−0.006▼

positions (Fig. (6.6)) confirms this negative trend. This is likely due to the nature of relevance judgment collection for the TREC Session Track data, where judgments are found for only the top ranked documents for the original subtopic. Based on the findings in this chapter, the specific subtopic of the query will change with every query, resulting in new documents being retrieved that were not evaluated by the TREC judges and so by default are considered non-relevant, thus reducing the metric scores.

What can be taken from these results is that when one comes across an impression which doesn't contain query terms, the next query is likely to be an improvement (regardless of whether the query term is *retained* or *removed*). Furthermore, in the converse scenario where query terms appear in all term sources, the next search ranking is likely to be worse.

The experiments conclude on an interesting final result, where it can be seen that when a term is *added* from a clicked document only (scenario 2), it leads to rankings with poorer IR scores. This is in spite of many of the findings that indicate that clicked documents are a rich source of *added* terms, that scenario 2 commonly occurs and that such reformulations lead to clicks 54.3% of the time. For example, the terms '`us government`' in Table (6.1) fall into scenario 2 for q_1 and are then *added* to q_2, the ranking of which leads to a click and an improvement in NERR

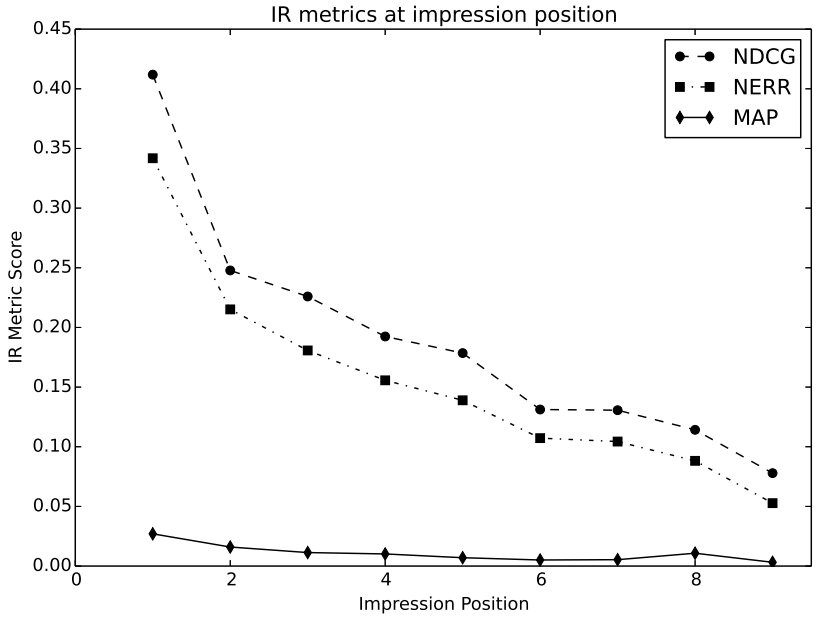


Figure 6.6: NERR@10, nDCG@10 and MAP scores for user created rankings at each impression position.

and nDCG, but not MAP, and then they are *removed*. Here, these terms represented a subtopic in the user's overall information need that was satisfied by their results before moving on. This result supports the argument that simply following the query reformulation behaviour of users does not necessarily lead to improved search systems, but through understanding the dynamic interactions with this methodology one can make more informed inferences.

6.7 Session Dynamics

This novel methodology and term-based approach to understanding query reformulation leads to some interesting, as well as expected, results. The results confirm that a user's query reformulation is largely made up of terms *retained* from their preceding query, with the remainder made up of a mix of terms discovered in the impression and externally sourced terms, although this can fluctuate throughout a session. However, one cannot expect to find all terms in add_{t+1} based on what is available in a query log, because users introduce terms based on their own cognitive processes, memory, external context or when changing their information need. For instance, in the example in Table (6.1), the final query contains the term '`law`' that

is not found in any of the term sources in the previous impression; it is clear from the table that this query is a departure from the topic and pattern of the previous queries. In such cases, techniques such as behavioural modeling, ontologies, contextual retrieval and topic modeling could be used to predict new terms to add, but this is beyond the scope of the work in this chapter.

While the terminology from the *DIR* framework is used throughout the chapter, here the links are made more explicit. In previous chapters the focus was on modeling the dynamics of a system responding to a user in an algorithm to improve retrieval over time, whether that be in a multi-page search or online learning to rank scenario. Inspired partly by the inferences that needed to be made in the UCB-DR evaluation in Chapter 5 to interpret the data in a session log, and the potential future goal of creating a session-based *DIR* ranking algorithm, in this chapter the goal has been to understand the user dynamics in a session search log.

In this case, each stage in the *DIR* framework represents an impression in a search session, and the actions available to the user are the term actions defined in this chapter, namely term *retention*, *removal* and *addition*. Additionally, the 8 term scenarios in Table (6.8) are merely an extension of this action space. The goal in this chapter has been to understand the unknown relevancies of the terms d and how they evolve over impressions, The findings suggest that terms belonging to different term sources and scenarios can have different relevancies to the user i.e. they are more or less likely to be used in the query reformulation. Aside from the observations of reformulation behaviour that can be gleaned from the search logs, the commonly used click observation is also used as a basis for evaluation and the effect of rank bias is also investigated. Following the work in this chapter, the next step would be to define a suitable static utility function that rewarded successful query reformulations, perhaps by using the statistics from Table (6.10), so that a dynamic utility could be defined and a query suggestion agent created.

6.8 Conclusion

In this chapter, a methodology for interpreting query reformulation behaviour based around the three term actions *retention*, *removal* and *addition* has been introduced. This technique was directly applied in an empirical analysis over TREC Session Track Data where the origin of terms used in query reformulations were analysed. From the results, the preceding query was identified as the main source but it was also shown that terms located in the impression itself were an additional source. It was found that adjacent queries in a session tended to be very similar but that there often was not a set of core teams that were used throughout, instead the core teams changed in the session as the information need changed.

The methodology was tested on well-understood findings in click model research and evidence was found of rank bias affecting reformulation behaviour. Following this, three user interaction based sources of terms were identified (and another based on dwell time was discarded) that were found in each impression and an experiment was conducted that tested from which sources users were able to locate terms to add to query reformulations. By matching query and impression terms in the term sources a number of possible user behaviour scenarios were defined to which a term could belong.

The difference in ranking quality of the term actions per scenario was measured to evaluate how good they were at not just predicting query reformulations, but effective ones. By interpreting the behaviour of the user for given scenarios and their corresponding effective actions, insight was gained that improved the understanding of a user's motivations for *retaining*, *removing* or *adding* terms.

Chapter 7

Conclusion

The work in this thesis addresses each of the five research questions posed in Section 1.5:

RQ1: *What is dynamic IR and can a framework for it be defined in the context of existing IR research?*

The background on dynamics itself and the motivation for its use in IR are covered in Chapters 1 and 2, including its three characteristics: *temporal dependency*, *user observation* and an *overall goal*. Further to this, a probabilistic model for solving problems in *DIR* is introduced in Chapter 3, where eight functional components are formally derived and presented in Table (3.1). These components were motivated by identifying features shared by static and interactive IR frameworks, in particular, by investigating the well-established *PRP* and *IIR-PRP* models. The *DIR* framework and its constituent elements are used throughout the remainder of this thesis for solving dynamic problems in IR.

RQ2: *Can the* DIR *framework be used in a practical setting? What are the benefits/drawbacks to its use? How does its use differ from other IR frameworks?*

The application of the *DIR* framework to practical ranking problems is demonstrated in Chapters 3, 4 and 5. Firstly, two *DIR* framework based algorithms are defined for document ranking in the multi-page search scenario. Settings for the framework components are carefully considered and discussed, resulting in practical algorithms for solving the MPS problem. The algorithm in Chapter 3 is compared against similarly derived algorithms based on the static and interactive frame-

works, while a different algorithm in Chapter 4 is evaluated against existing ranking and retrieval techniques such as the Rocchio and MMR algorithms.

These evaluations lead to interesting results regarding the effectiveness of ranking with the *DIR* framework, in particular they demonstrate that static or interactive rankings may be more beneficial in some cases, but in others (for instance when diversification is needed), dynamic ranking is preferable. Drawbacks to the framework are also discovered and discussed with regard to **RQ4**. These issues are further addressed in Chapter 5, where two algorithms for the online learning to rank problem are presented. In one of the algorithms, dynamic modeling leads to the exploration of documents, whereas in the other diversification occurs. A common finding throughout this thesis has been that it is important to find the right amount of exploration (or diversification) to pursue in *DIR*, if there is too much then the search performance suffers.

RQ3: *In what ways can reinforcement learning methods, sampling and other approximations be used to minimise the impact of complexity in the* DIR *framework?*

In Chapter 3, the framework is likened to the POMDP formulation, resulting in the use of the Bellman equation to define the dynamic utility function. A consequence of this type of model is the computational complexity of solving the Bellman equation when a large number of states and actions is considered, which is often the case in *DIR*. As such, methods including the sequential ranking decision and Monte-Carlo sampling are used throughout the algorithms in Chapters 3, 4 and 5. In Chapter 5 this is extended further by making use of a multi-armed bandit, index-based formulation, derived from the application of expectation maximisation to document relevance. This simplifies the *DIR* formulation into two efficient ranking algorithms.

RQ4: *How should IR problems in the* DIR *framework be evaluated? What are the problems encountered when evaluating* DIR *algorithms?*

A recurring problem throughout this thesis has been the evaluation of the described algorithms. Conventional IR metrics are used throughout; for instance,

while in Chapter 3 metrics are measured over individual pages of results, in Chapter 4 they are measured over a combination of results pages. Session based metrics were also explored in Chapter 3, but with poor results. In Chapter 5, the lack of an appropriate live user testing system resulted in the algorithms being evaluated through simulations and over search log data, with mixed results. *DIR* algorithms are by their nature responsive and adaptive to users, and so without actual users to interact with a *DIR* system it can be difficult to conduct a proper and unbiased evaluation. Further, few publicly available search logs contain interaction data such as clicks. These problems led to the work conducted in Chapter 6, where the dynamics in a search log were directly extracted and investigated. Nonetheless, the different evaluation methodologies and metrics used throughout this thesis are indicative of what is an open problem in dynamic information retrieval research.

RQ5: *What does the* DIR *framework reveal about the interactions that occur in a dynamic system? Can search logs from a dynamic system be used to understand dynamic interactions?*

From the experiment using Yandex data in Chapter 5 and the full TREC Session Track search log analysis conducted in Chapter 6, interesting insights are obtained about the dynamic search behaviour of users. For instance, it was found that the core terms of a query tend to change throughout the course of a session, signifying the shifting of query intent. Further to this, the terms from highly ranked snippets are more likely to feature in the user's next query, rather than the terms from snippets near to clicked documents. These results demonstrate that the *DIR* framework can be used to extract interesting user insights from search log datasets.

7.1 Future Work

In this thesis, only a few of the potential settings of the *DIR* framework components have been explored. Different problem settings in dynamic IR result in different definitions for the relevance, actions, observations and so on. For instance, in Chapters 5 and 6 the focus is on the user as the agent, rather than the search system, requiring a different understanding of the framework to the previous two

chapters. Even within the well-understood problem of ranking and retrieval, there exists a wide range of IR research that can be utilised in the framework settings. For instance, different click models can be incorporated into the probability of observation distribution, other utility models could be used for the static utility, and the relevance and its covariance could be modeled using a distribution other than a Gaussian.

There is room for improvement in the multi-page search algorithms in Chapters 3 and 4. One improvement would be to build a classifier that could learn for which queries results diversification is beneficial, otherwise the first page is ranked according to the *PRP* [80]. With some modifications the algorithms could also be tested in the session search scenario, as was demonstrated by Luo et al. [130]. The multi-armed bandit based formulation from Chapter 5 could also be applied to the multi-page search scenario and evaluated against the direct *DIR* framework implementations in this thesis. Further to this, other reinforcement learning techniques could be utilised in dynamic online learning to rank, as well as other multi-armed bandit algorithm variants such as the duelling bandit [128] or the interleaving techniques used in online learning to rank [158].

The work in Chapter 6 could be extended by further breaking down an impression into new term sources, such as snippet and document title or document components such as headers and paragraph text. Features such as rank and impression position or click order could be used to separate the current term sources and increase the number of scenarios. An n-gram model would require different similarity measures but would allow for more accurate phrase matching and new term actions (such as phrase rearrangement, splitting etc.). Term sources from non-adjacent impressions could also help to improve the overall model, and other implicit user measures (such as mouse tracking or reading level) could prove a good differentiator of term source similarity. This work could then be extended to define a query suggestion agent for session search that uses a dynamic model built up from the occurrence of terms in the preceding impressions and queries. Likewise, an accurate model of the user dynamics could aid in improving user behaviour models

and allow for the use of session search logs in the evaluation of dynamic techniques.

In addition, the recall-oriented nature of dynamic IR, an important factor in research on explorative and diversified search, is insufficiently investigated in this thesis. As reported in the experiment findings, the metric scores of individual pages may suffer in dynamic IR so that the overall scores can be optimised. This can be described by the familiar balance between precision and recall, where increased recall is a result of diversified rankings. Subsequently, recall oriented datasets, such as in patent and legal search, may have been useful for the evaluation in this thesis. Further to this, new technologies such as mobile search, which by its nature benefits from streamed search results and a reduced number of user interactions, are potential practical applications of dynamic IR that are not explored in this thesis.

Ultimately, a key challenge facing the design of algorithms under the *DIR* framework is in their evaluation. While industrial practitioners in IR may have access to a live search system and users on which to perform A/B tests, these resources are not typically available to academics on a scale larger than expensive user studies. The inherent interactivity of *DIR* systems means that evaluating over offline test collections will always result in compromises, which has been the case in this thesis. In recent years, some headway has been made in the use of offline datasets for the evaluation of online learning techniques [175, 176], in particular the work on causal inference which offers promising advances [177]. In addition, the recent Living Labs for IR[1] project allows academics to create document rankings for live users, a useful research tool for evaluating dynamic techniques.

Finally, while the proposed definition of dynamic information retrieval and the probabilistic framework defined in this thesis is useful for helping to design dynamic algorithms and understand the benefits and limitations of *DIR*, it does not directly identify solutions for solving such problems. Typically when using a Markovian solution such as an MDP, a stationary solution can be found by taking the limit of the dynamic utility. This solution is usually easier to solve than by searching over the action and state space as is the case in the algorithms in this thesis. Here, a

[1] `http://living-labs.net/`

stationary solution is not pursued, instead the framework and its applications are thoroughly explored. The ultimate, long-term objective in the area of dynamic information retrieval is to find a stationery solution to the dynamic utility function and to understand how to apply it to problems in IR.

Appendix A

Related Publications

The following publications form the main part of this thesis:

1. Hui Yang, Marc Sloan, and Jun Wang. *Dynamic Information Retrieval Modeling*. Morgan & Claypool, 2015

2. Hui Yang, Marc Sloan, and Jun Wang. Dynamic information retrieval modeling (tutorial). SIGIR '14, page 1290. ACM, 2014

3. Hui Yang, Marc Sloan, and Jun Wang. Dynamic information retrieval modeling (tutorial). In *WSDM '15*, pages 409–410. ACM Press, 2015

4. Marc Sloan and Jun Wang. Dynamic information retrieval: Theoretical framework and application. ICTIR'15. ACM, 2015

5. Xiaoran Jin, Marc Sloan, and Jun Wang. Interactive exploratory search for multi page search results. In *WWW '13*, pages 655–666, 2013

6. Marc Sloan and Jun Wang. Iterative expectation for multi period information retrieval. In *WSDM Workshop on Web Search Click Data*. WSCD, 2013

7. Marc Sloan and Jun Wang. Dynamical information retrieval modelling: a portfolio-armed bandit machine approach (poster). WWW '12, pages 603–604. ACM, 2012

8. Marc Sloan, Hui Yang, and Jun Wang. A term-based methodology for query reformulation understanding. *Information Retrieval Journal*, 18(2):145–165,

2015

In addition, there were a number of publications completed during the formation of this thesis that concern related topics such as contextual, session-based retrieval and computational advertising:

1. Milad Shokouhi, Marc Sloan, Paul N. Bennett, Kevyn Collins-Thompson, and Siranush Sarkizova. Query suggestion and data fusion in contextual disambiguation. WWW '15, pages 971–980, 2015

2. Shuai Yuan, Ahmad Zainal Az Abidin, Marc Sloan, and Jun Wang. Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users. *CoRR*, abs/1206.1:1–44, 2012

Appendix B

Glossary

Action

An *action* is taken by a dynamic agent to affect their environment. In dynamic information retrieval, an action is typically a ranking of documents.

Agent

The *agent* (or dynamic agent) operates in an environment by making actions and observing feedback signals (see Fig. (1.1)).

Bellman equation

The *Bellman equation* is a recursive utility function used in the dynamic programming solution to an MDP [41].

BM25

BM25 is a well-established method for scoring document relevance using term frequencies [2].

Click

A *click* (or *clickthrough*) is an observation of a user clicking on a link, usually in a list of search results. Clicks are often recorded in search log datasets.

Click model

A *click model* is a probabilistic model that represent a user's typical behaviour on a search page.

Co-click

A *co-click* occurs when two documents in the same ranked list are clicked.

Co-show

A *co-show* occurs when two documents are displayed together in the same ranked list.

Control theory

Control theory is the mathematics of closed feedback looping systems that maintain some form of equilibrium or state [10].

Cosine similarity

The *cosine similarity* is a measure of the angle between two term vectors in the vector space model.

DCG

The *Discounted Cumulative Gain* (DCG) is a measure of the quality of a ranking of documents [184].

DES

The *Dynamic Exploratory Search* (DES) algorithm from Chapter 4, which is a specific implementation of the dynamic IR framework for multi-page search.

DIR

A *Dynamic Information Retrieval* (DIR) system is one which changes or adapts over time. In this thesis, dynamic IR is defined as the modeling of adaptive, responsive, goal-oriented information retrieval systems. The *DIR* framework is is a probabilistic model for solving problems in dynamic IR.

DIR-MPS

The *Dynamic IR - Multi-Page Search* algorithm from Chapter 3, which demonstrates how the dynamic IR framework can be applied to the problem of multi-page search.

Document

A *document* is a unit of information contained in a corpus, for instance, a webpage in web search.

Dynamic programming

Dynamic programming is the process of breaking down a difficult problem into sub-problems that can be solved independently, then their solutions combined to solve the main problem.

Dynamic utility

The *dynamic utility* is the recursive utility function in the dynamic IR framework that can be used to solve *DIR* problems.

ERR

The *Expected Reciprocal Rank* (ERR) is the inverse of the expected rank position of the first relevant document in a ranked list.

Explicit relevance feedback

In *explicit relevance feedback*, the user is directly solicited for information regarding the relevance of information items, or else their information need preferences.

Feedback

Feedback is a contextual relevance signal received by a user. It can be *explicit* (such as a rating) or *implicit* (such as a click).

Gittens index

The *Gittens index* value represents the expected reward for playing an arm in a multi-armed bandit until a termination step, and so for each time step, the arm with the highest Gittens index value is the arm chosen to play [43].

Interactive Information Retrieval

Interactive information retrieval research explores the complex sequence of interactions a user may have with a search ranking [137], largely motivated by the contradictory results found from conventional Cranfield style evaluation [138] and observational user studies [139].

IIR-PRP

The *Probability Ranking Principle for Interactive Information Retrieval* (IIR-PRP) is a framework for ranking in static IR, whereby documents are ranked according to user effort and benefit and are dependent on previously ranked documents [129].

Implicit relevance feedback

In *implicit relevance feedback*, signals observed from natural user interactions with the search system are unobtrusively recorded and interpreted as signals of relevance [102].

Impression

An *impression* is a single instance of a search in a search log. It consists of a query, it's ranked list of results, and may also include clicks, documents and snippets.

Information need

An *information need* defines information that a user is seeking using an IR system, which can be interpreted as a task or topic.

IR

Information Retrieval (IR) is the study of obtaining information resources relevant to an information need.

Jaccard similarity

The *Jaccard similarity* is a measure of the overlap between two sets.

Learning to rank

Learning to rank is the application of supervised machine learning algorithms to the space of search tasks.

MAB

A *Multi-Armed Bandit* (MAB) is a reinforcement learning algorithm that explores potential actions and exploits actions known to give high rewards.

MAP

The *Mean Average Precision* (MAP) is a measure of the average precision (the number of relevant documents) across queries in a dataset.

MDP

A *Markov Decision Process* (MDP) is a mathematical framework for making optimal decisions. It typically consists of a set of states, actions, a transition probability distribution and a reward function.

MMR

The *Maximal Marginal Relevance* (MMR) algorithm is a diversification technique where documents are re-ranked to take into account dis-similarity [17].

MPS

The *Multi-Page Search* (MPS) scenario is a search problem in *DIR* where rankings must be generated for multiple pages of search results for an individual query.

NDCG

The *normalised Discounted Cumulative Gain* (nDCG) is the normalised DCG score.

Observation

An *observation* is a feedback signal from a dynamic agent's environment.

Observation likelihood function

In a POMDP framework, agents cannot determine which state they are currently in directly, but they can make an observation which gives some indication of the state they are in, given by the *observation likelihood function*.

Online learning to rank

Online learning to rank is described 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"* [18].

Path-discount

The *path-discount* function gives a weight to the future stages in a dynamic system.

PAB

The *Portfolio-Armed Bandit* (PAB) algorithm from Chapter 5 combines portfolio theory with a multi-armed bandit algorithm so that diverse rankings of documents can be learned over time.

POMDP

A *Partially Observable Markov Decision Process* (POMDP) is an MDP where the agent in unsure of which state it is in. As well as the set of states, actions, the transition probability distribution and reward function, it also comprises a set of observations and an observation likelihood function.

Probability of relevance

The *probability of relevance* is a value between 0 and 1 representing the probability that a document is relevant.

PRP

The *Probability Ranking Principle* (PRP) is a framework for ranking in static IR, whereby documents are ranked in decreasing order of their probability of relevance [5].

Pseudo relevance feedback

Pseudo relevance feedback systems simulate explicit relevance feedback by assuming that the top k ranked documents in a search ranking are relevant, then conventional relevance feedback is performed to improve search results [106, 107].

Query

A *query* is a collection of terms used by a user to represent an information need during information retrieval.

Query reformulation

A *query reformulation* occurs when a user searches for a new query during session search.

Regret

Regret is the commonly used evaluative framework for multi-armed bandit algorithms, defined as the difference between the sum of rewards of the optimal sequence of arms versus those chosen by the multi-armed bandit algorithm.

RL

A *Reinforcement Learning* (RL) algorithm is a machine learning algorithm that can learn an optimal sequence of actions by making observations on its environment and determining rewards and penalties.

Relevance

The *relevance* of a document is a hidden measure of whether it satisfies a given information need.

Relevance update function

The *relevance update function* τ updates the relevance score based on a dynamic agent's action and an observation. Related to the transition probability distribution in an MDP.

Reward

In an MDP, the *reward* function defines the reward that is given to the agent given their current state, their previous state and the action that is taken between them.

Rocchio

The *Rocchio* algorithm is a relevance feedback technique that uses the vector space model to alter a user's query based on documents they have judged to be relevant and non-relevant [16].

SAP

The *Session-Average Precision* (sAP) metric, where the average precision is calculated over all possible paths of interactions in a search session [132].

SDCG

The *Session-DCG* (sDCG) metric calculates the DCG score across all ranked documents in a session [141].

Search log

A *search log* (or query log) is a dataset consisting of recorded search impressions, including their queries, displayed URLs, clicks, dwell time etc.

Sequential ranking decision

The *sequential ranking decision* is an approximation technique, where instead of searching the space of all document rankings for an optimal one, the optimal document for each ranking position is found rank by rank instead.

SERP

The *Search Engine Results Page* (SERP) is a webpage that displays search results to a user.

Session

A search *session* is a period of time in which a user issues multiple queries consecutively, usually in pursuit of an individual information need.

Stage

A *stage* represents an interaction with the search system that is distinct from other interactions but belongs to the same search task, for example a sequence of impressions in session search.

State

In an MDP framework, the *state* describes the status that the agent is in at a given moment in time.

Static IR

A *static IR* framework is one which models single user interactions, or else multiple independent interactions of different search intents. A typical application would be an ad hoc ranking and retrieval system.

Static utility

The *static utility* function gives value to an action based on its probability of relevance by modeling the benefit of the action to the user.

Temporal dependency

Temporal dependency is introduced into a dynamic system when the relevance score at a particular stage depends on the previous stage.

Transition probability distribution

In an MDP framework, the *transition probability distribution* describes the probability of transitioning to a new state given the current state and the action.

TREC

The annual *Text REtrieval Conference* (TREC) series, known for publishing open competitions in areas of research in IR with accompanying datasets.

UCB

The *Upper Confidence Bound* (UCB) algorithm is a well-known index based multi-armed bandit algorithm that balances the exploration and exploitation of playing arms [118].

UCB-DR

The *Upper Confidence Bound - Dynamic Ranking* (UCB-DR) algorithm from Chapter 5 is a dynamic online learning to rank algorithm that is based on the UCB algorithm and assigns an index score for each document based on its estimated probability of relevance and an exploration element.

User

A *user* is an agent that uses a search system, issuing queries and examining and clicking on documents.

Utility

A *utility* function measures the value of an action by an agent.

Bibliography

[1] Evangelos Kanoulas, Ben Carterette, Mark Hall, Paul Clough, and Mark Sanderson. Overview of the TREC 2013 session track. In *TREC'13*.

[2] Robertson Stephen and Zaragoza Hugo. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

[3] Xing Wei and W Bruce Croft. LDA-based document models for ad-hoc retrieval. SIGIR '06, pages 178–185.

[4] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, March 2009.

[5] Stephen Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[6] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services Series. Morgan & Claypool, 2009.

[7] Marc Sloan, Hui Yang, and Jun Wang. A term-based methodology for query reformulation understanding. *Information Retrieval Journal*, 18(2):145–165, 2015.

[8] Tamas Jambor, Jun Wang, and Neal Lathia. Using control theory for stable and efficient recommender systems. WWW '12, pages 11–20. ACM, 2012.

[9] Shuai Yuan and Jun Wang. Sequential selection of correlated ads by POMDPs. CIKM '12, pages 515–524. ACM, 2012.

[10] Oliver M. O'Reilly. *Engineering Dynamics: A Primer*. Springer, 2001.

[11] Joseph J. Distefano, Allen R. Stubberud, and Ivan J. Williams. *Schaum's Interactive Feedback and Control Systems*. McGraw-Hill, 1994.

[12] Robert J. Sternberg. *Handbook of Intelligence*. Cambridge University Press, 2000.

[13] Hui Yang, Marc Sloan, and Jun Wang. Dynamic information retrieval modeling (tutorial). SIGIR '14, page 1290. ACM, 2014.

[14] Jay M Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98*.

[15] Gerard Salton, Andrew Wong, and Chungshu S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[16] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System*, pages 313–323. Prentice-Hall, 1971.

[17] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. SIGIR '98, pages 335–336. ACM, 1998.

[18] Katja Hofmann. *Fast and Reliable Online Learning to Rank for Information Retrieval*. PhD thesis, University of Amsterdam, 2013.

[19] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. ICML '07, pages 129–136. ACM, 2007.

[20] Thorsten Joachims. Optimizing search engines using clickthrough data. KDD '02, pages 133–142. ACM, 2002.

[21] Christopher J. C. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with non-smooth cost functions. In *Advances in Neural Information Processing Systems 19*, pages 193–200. MIT Press, 2007.

[22] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Balancing exploration and exploitation in learning to rank online. ECIR'11, pages 251–263, 2011.

[23] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. ICML '08, pages 784–791. ACM, 2008.

[24] Filip Radlinski and Nick Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*, pages 245–254.

[25] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW '10*, pages 661–670.

[26] Massimiliano Ciaramita, Vanessa Murdock, and Vassilis Plachouras. Online learning from click data for sponsored search. WWW '08, 2008.

[27] Bernard J. Jansen, Amanda Spink, and Jan Pedersen. A temporal comparison of AltaVista web searching: Research articles. *Journal of the American Society for Information Science and Technology*, 56(6):559–570, 2005.

[28] Filip Radlinski and Thorsten Joachims. Query chains: Learning to rank from implicit feedback. KDD '05, pages 239–248. ACM, 2005.

[29] Ruihua Song, Zhenxiao Luo, Jian-Yun Nie, Yong Yu, and Hsiao-Wuen Hon. Identification of ambiguous queries in web search. *Information Processing & Management*, 45(2):216–229, 2009.

[30] Gary Marchionini. Exploratory search: From finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[31] Lydia B Chilton and Jaime Teevan. Addressing people's information needs directly in a web search result page. WWW '11, pages 27–36, 2011.

[32] Marc-Allen Cartright, Ryen W. White, and Eric Horvitz. Intentions and attention in exploratory health search. In *SIGIR '11*, pages 65–74, 2011.

[33] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *SIGIR '08*, pages 163–170, 2008.

[34] Dongyi Guan, Sicong Zhang, and Hui Yang. Utilizing query change for session search. SIGIR '13, pages 453–462. ACM, 2013.

[35] Stephen Robertson, Melvin E. Maron, and William S. Cooper. Probability of relevance: A unification of two competing models for document retrieval. *Information Technology: Research and Development*, 1(1):1–21, 1982.

[36] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. SIGIR '08, pages 659–666. ACM, 2008.

[37] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168, 2005.

[38] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. WSDM '08, pages 87–94. ACM, 2008.

[39] William S. Cooper. On selecting a measure of retrieval effectiveness. *Journal of the American Society for Information Science*, 24(2):87–100, 1973.

[40] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. SIGIR' 09, pages 115–122. ACM, 2009.

[41] Richard Bellman. A Markovian decision process. *Indiana University Mathematics Journal*, 6(4):679–684, 1957.

[42] Richard Bellman. *Dynamic Programming*. Dover Books on Mathematics. Princeton University Press, 2003.

[43] John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B (Methodological)*, 41(2):148–177, 1979.

[44] Stephen E. Robertson and Karen Spärck Jones. Document retrieval systems. chapter Relevance Weighting of Search Terms, pages 143–160. Taylor Graham Publishing, 1988.

[45] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.

[46] Azin Ashkan, Charles L. Clarke, Eugene Agichtein, and Qi Guo. Classifying and characterizing query intent. ECIR '09, pages 578–586. Springer-Verlag, 2009.

[47] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in Web search. WWW '05, pages 391–400. ACM, 2005.

[48] Ben Carterette and Rosie Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *Advances in Neural Information Processing Systems 20*, pages 217–224. 2008.

[49] Jaap Kamps, Marijn Koolen, and Andrew Trotman. Comparative analysis of clicks and judgments for ir evaluation. WSCD '09, pages 80–87. ACM, 2009.

[50] Falk Scholer, Milad Shokouhi, Bodo Billerbeck, and Andrew Turpin. Using clicks as implicit judgments: Expectations versus observations. ECIR'08, chapter 6, pages 28–39. Springer-Verlag, 2008.

[51] Milad Shokouhi, Falk Scholer, and Andrew Turpin. Investigating the effectiveness of clickthrough data for document reordering. ECIR'08, pages 591–595. Springer-Verlag, 2008.

[52] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems*, 25(2), 2007.

[53] Thorsten Joachims, Laura Granka, Bing Pang, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. SIGIR '05, pages 154–161, 2005.

[54] Charles L. A. Clarke, Eugene Agichtein, Susan Dumais, and Ryen W. White. The influence of caption features on clickthrough patterns in web search. SIGIR '07, pages 135–142. ACM, 2007.

[55] Jiepu Jiang, Daqing He, and James Allan. Searching, browsing, and clicking in a search session: Changes in user behavior by task and over time. SIGIR '14, pages 607–616. ACM, 2014.

[56] Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. WSDM '09, pages 124–131. ACM, 2009.

[57] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. Click chain model in web search. WWW '09, pages 11–20. ACM, 2009.

[58] Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. WWW '09, pages 1–10. ACM, 2009.

[59] Yin He and Kuansan Wang. Inferring search behaviors using partially observable markov model with duration (POMD). WSDM '11, pages 415–424. ACM, 2011.

[60] Guido Zuccon, Leif A. Azzopardi, and Keith van Rijsbergen. The quantum probability ranking principle for information retrieval. In *Advances in Information Retrieval Theory*, volume 5766, pages 232–240. Springer Berlin Heidelberg, 2009.

[61] Harr Chen and David R Karger. Less is more: Probabilistic models for retrieving fewer relevant documents. SIGIR '06, pages 429–436. ACM, 2006.

[62] Guido Zuccon, Leif Azzopardi, Dell Zhang, and Jun Wang. Top-k retrieval using facility location analysis. ECIR'12, pages 305–316. Springer-Verlag, 2012.

[63] Leif Azzopardi. The economics in interactive information retrieval. SIGIR '11, pages 15–24. ACM, 2011.

[64] Jiyun Luo, Sicong Zhang, and Hui Yang. Win-win search: Dual-agent stochastic game in session search. SIGIR '14, 2014.

[65] Mikhail Ageev, Qi Guo, Dmitry Lagun, and Eugene Agichtein. Find it if you can: A game for modeling different types of web search success using interaction data. SIGIR '11. ACM, 2011.

[66] Nikolay Archak, Anindya Ghose, and Panagiotis G. Ipeirotis. Show me the money!: Deriving the pricing power of product features by mining consumer reviews. KDD '07, pages 56–65. ACM, 2007.

[67] Leif Azzopardi. Modelling interaction with economic models of search. SIGIR '14, pages 3–12. ACM, 2014.

[68] Harry Markowitz. A simplex method for the portfolio selection problem. Cowles Foundation Discussion Papers 27, Cowles Foundation for Research in Economics, Yale University, 1957.

[69] Kevyn Collins-Thompson. *Robust Model Estimation Methods for Information Retrieval*. PhD thesis, Carnegie Mellon University, December 2008.

[70] Lidan Wang, Paul N. Bennett, and Kevyn Collins-Thompson. Robust ranking models via risk-sensitive optimization. SIGIR '12, pages 761–770. ACM, 2012.

[71] James C. Maxwell and William D. Niven. *The Scientific Papers of James Clerk Maxwell*. Dover Publications, 2003.

[72] Seth B. Anderson. *A look at handling qualities of Canard configurations*. NASA technical memorandum. 1986.

[73] Volnei A. Pedroni. *Digital Electronics and Design with VHDL*. Elsevier Science, 2008.

[74] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. *Feedback control of computing systems*. John Wiley & Sons, 2004.

[75] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence : MDPs, beyond MDPs and applications*. ISTE Hoboken, NJ, 2010.

[76] Michael I. Jordan. Computational aspects of motor control and motor learning. In *Handbook of Perception and Action: Motor Skills*. Academic Press, 1993.

[77] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[78] Burr Settles. Active learning literature survey. Computer sciences technical report, 2009.

[79] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[80] Milad Shokouhi, Marc Sloan, Paul N. Bennett, Kevyn Collins-Thompson, and Siranush Sarkizova. Query suggestion and data fusion in contextual disambiguation. WWW '15, pages 971–980, 2015.

[81] Jiannong Cao, Kwok Ming Chan, Geofffrey Yu-Kai Shea, and Minyi Guo. Location-aware information retrieval for mobile computing. In *Embedded and Ubiquitous Computing*, volume 3207, pages 450–459. Springer Berlin Heidelberg, 2004.

[82] Carsten Eickhoff, Kevyn Collins-Thompson, Paul N. Bennett, and Susan Dumais. Personalizing atypical web search sessions. In *WSDM '13*, pages 285–294, 2013.

[83] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Xuanhui Wang. Click shaping to optimize multiple objectives. KDD '11, pages 132–140. ACM, 2011.

[84] Amanda Spink, Minsoo Park, Bernard J. Jansen, and Jan Pedersen. Multitasking during web search sessions. *Information Processing & Management*, 42(1):264–275, 2006.

[85] Anagha Kulkarni, Jaime Teevan, Krysta M. Svore, and Susan T. Dumais. Understanding temporal query dynamics. WSDM '11, pages 167–176. ACM, 2011.

[86] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. KDD '08, pages 875–883. ACM, 2008.

[87] Milad Shokouhi. Learning to personalize query auto-completion. SIGIR '13, pages 103–112. ACM, 2013.

[88] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. A large-scale study of the evolution of web pages. WWW '03, pages 669–678. ACM, 2003.

[89] Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas. The web changes everything: Understanding the dynamics of web content. WSDM '09, pages 282–291. ACM, 2009.

[90] Taesun Moon and Jason Baldridge. Part-of-speech tagging for middle English through alignment and projection of parallel diachronic texts. EMNLP-CoNLL, pages 390–399. Association for Computational Linguistics, 2007.

[91] Uri Hanani, Bracha Shapira, and Peretz Shoval. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, 11(3):203–259, 2001.

[92] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. KDD '09, pages 497–506. ACM, 2009.

[93] Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. WSDM '11, pages 177–186. ACM, 2011.

[94] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. Towards recency ranking in web search. WSDM '10, pages 11–20. ACM, 2010.

[95] Larry Heck. The conversational web. IEEE Workshop on Spoken Language Technology, 2012.

[96] Peter W. Foltz and Susan T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1992.

[97] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. SIGIR '10, pages 363–370. ACM, 2010.

[98] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.

[99] Ryen W. White, Ian Ruthven, and Joemon M. Jose. A study of factors affecting the utility of implicit relevance feedback. SIGIR '05, pages 35–42. ACM, 2005.

[100] Ryen W. White and Ian Ruthven. A study of interface support mechanisms for interactive information retrieval. *Journal of the American Society for Information Science and Technology*, 57(7):933–948, 2006.

[101] Amanda Spink, Bernard J. Jansen, and Cenk H. Ozmultu. Use of query reformulation and relevance feedback by Excite users. *Internet Research: Electronic Networking Applications and Policy*, 10(4):317–328, 2000.

[102] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[103] Masahiro Morita and Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. SIGIR '94, pages 272–281. Springer-Verlag New York, Inc., 1994.

[104] Youngho Kim, Ahmed Hassan, Ryen W. White, and Imed Zitouni. Modeling dwell time to predict click-level satisfaction. WSDM '14, pages 193–202. ACM, 2014.

[105] Jeremy Goecks and Jude Shavlik. Learning users' interests by unobtrusively observing their normal behavior. IUI '00, pages 129–132. ACM, 2000.

[106] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR '08*, pages 243–250. ACM.

[107] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. SIGIR '96, pages 4–11, 1996.

[108] Chris Buckley. Automatic query expansion using SMART : TREC 3. TREC-3, pages 69–80.

[109] Van Dang and W. Bruce Croft. Query reformulation using anchor text. WSDM '10, pages 41–50. ACM, 2010.

[110] Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, June 2003.

[111] Holger Bast and Ingmar Weber. Type less, find more: Fast autocompletion search with a succinct index. SIGIR '06, pages 364–371. ACM, 2006.

[112] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. EDBT'04, pages 588–596. Springer-Verlag, 2004.

[113] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. ICML '97, pages 143–151, 1997.

[114] Robert M. Blumenthal and Ronald K. Getoor. *Markov Processes and Potential Theory*. Academic Press, 1968.

[115] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge University, 1989.

[116] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[117] Tze L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[118] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[119] Rajeev Agrawal. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):pp. 1054–1078, 1995.

[120] Satyen Kale, Lev Reyzin, and Robert Schapire. Non-stochastic bandit slate problems. In *Advances in Neural Information Processing Systems 23*, pages 1045–1053. 2010.

[121] Lifeng Lai, H. El Gamal, Hai Jiang, and H.V. Poor. Cognitive medium access: Exploration, exploitation, and competition. *Mobile Computing, IEEE Transactions on*, 10(2):239–253, 2011.

[122] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. ICML '07, pages 721–728. ACM, 2007.

[123] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems 27*, pages 199–207. 2014.

[124] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine Learning*, 80(2-3):245–272, 2010.

[125] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Neural Information Processing Systems*, pages 273–280, 2008.

[126] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pages 817–824. 2008.

[127] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The K-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[128] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. ICML '09, pages 1201–1208, 2009.

[129] Norbert Fuhr. A probability ranking principle for interactive information retrieval. *Information Retrieval*, 11(3):251–265, 2008.

[130] Jiyun Luo, Sicong Zhang, Xuchu Dong, and Hui Yang. Designing states, actions, and rewards for using POMDP in session search. In *Advances in Information Retrieval*, pages 526–537. Springer International Publishing, 2015.

[131] Edward J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted cost. *Operations Research*, 26(2):282–304, 1978.

[132] Evangelos Kanoulas, Ben Carterette, Paul D. Clough, and Mark Sanderson. Evaluating multi-query sessions. SIGIR '11, pages 1053–1062. ACM, 2011.

[133] Jin Young Kim, Mark Cramer, Jaime Teevan, and Dmitry Lagun. Understanding how people interact with web search results that change in real-time using implicit feedback. CIKM '13, pages 2321–2326. ACM, 2013.

[134] Xiaoran Jin, Marc Sloan, and Jun Wang. Interactive exploratory search for multi page search results. In *WWW '13*, pages 655–666, 2013.

[135] Jun Wang and Jianhan Zhu. On statistical analysis and optimization of information retrieval effectiveness metrics. SIGIR '10, pages 226–233, 2010.

[136] William S. Cooper. The inadequacy of probability of usefulness as a ranking criterion for retrieval system output. *University of California, Berkeley*, 1971.

[137] Ian Ruthven. Interactive information retrieval. *ARIST*, 42(1):43–91, 2008.

[138] Cyril Cleverdon and Michael Kean. Factors determining the performance of indexing systems. Aslib Cranfield Research Project, 1968.

[139] William Hersh, Andrew Turpin, Susan Price, Benjamin Chan, Dale Kramer, Lynetta Sacherek, and Daniel Olson. Do batch and user evaluations give the same results? SIGIR '00, pages 17–24. ACM, 2000.

[140] Yiming Yang and Abhimanyu Lad. Modeling expected utility of multi-session information distillation. In *Advances in Information Retrieval Theory*, pages 164–175. Springer Berlin Heidelberg, 2009.

[141] Kalervo Järvelin, Susan L. Price, Lois M. L. Delcambre, and Marianne Lykke Nielsen. Discounted cumulated gain based evaluation of multiple-query IR sessions. ECIR'08, pages 4–15. Springer-Verlag, 2008.

[142] Bernard J. Jansen and Amanda Spink. How are we searching the world wide web?: A comparison of nine search engine transaction logs. *Information Processing & Management*, 42(1):248–263, 2006.

[143] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. WSDM '09, pages 5–14. ACM, 2009.

[144] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. CIKM '09, pages 621–630. ACM, 2009.

[145] Kuansan Wang, Nikolas Gloy, and Xiaolong Li. Inferring search behaviors using partially observable Markov (POM) model. WSDM '10, pages 211–220. ACM, 2010.

[146] Ryen W. White, Ian Ruthven, Joemon M. Jose, and C. J. Van Rijsbergen. Evaluating implicit feedback models using searcher simulations. *ACM Transactions on Information Systems*, 23(3):325–361, 2005.

[147] Mark D Smucker and Charles L. A. Clarke. Time-based calibration of effectiveness measures. SIGIR '12, pages 95–104. ACM, 2012.

[148] Marco Ferrante, Nicola Ferro, and Maria Maistro. Injecting user models and time into precision via Markov chains. SIGIR '14, pages 597–606. ACM, 2014.

[149] Michael L. Littman. The Witness algorithm: Solving partially observable Markov decision processes. Technical report, 1994.

[150] Bill Kules and Robert Capra. Visualizing stages during an exploratory search session. HCIR '11, 2011.

[151] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. CIKM '05, pages 824–831. ACM, 2005.

[152] Christina Brandt, Thorsten Joachims, Yisong Yue, and Jacob Bank. Dynamic ranked retrieval. WSDM '11, pages 247–256. ACM, 2011.

[153] Mark Cramer, Mike Wertheim, and David Hardtke. Demonstration of improved search result relevancy using real-time implicit relevance feedback. In *Understanding the user - workshop in conjuction with SIGIR'09*.

[154] Ellen M Voorhees. The cluster hypothesis revisited. SIGIR '85, pages 188–196, 1985.

[155] Zuobing Xu and Ram Akella. Active relevance feedback for difficult queries. CIKM '08, pages 459–468. ACM, 2008.

[156] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and WeiGuo Fan. Optimizing web search using web click-through data. CIKM '04, pages 118–126. ACM, 2004.

[157] Pannagadatta K Shivaswamy and Thorsten Joachims. Online learning with preference feedback. *CoRR*, 2011.

[158] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. Reusing historical interaction data for faster online learning to rank for IR. WSDM '13, pages 183–192. ACM, 2013.

[159] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. FOCS '95, pages 322–331, 1995.

[160] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. SIGIR '06, pages 3–10. ACM, 2006.

[161] Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. In *International Conference on Systems Integration*, 1997.

[162] Yuchen Zhang, Dong Wang, Gang Wang, Weizhu Chen, Zhihua Zhang, Botao Hu, and Li Zhang. Learning click models via probit Bayesian inference. CIKM '10, pages 439–448. ACM, 2010.

[163] David J. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII*, pages 1–198, 1983.

[164] Ryen W. White and Steven M. Drucker. Investigating behavioral variability in web search. WWW '07, pages 21–30, 2007.

[165] Khamsum Kinley, Dian Tjondronegoro, Helen Partridge, and Sylvia Edwards. Human-computer interaction: The impact of users' cognitive styles on query reformulation behaviour during web searching. OzCHI '12, pages 299–307. ACM, 2012.

[166] Evangelos Kanoulas, Ben Carterette, Mark Hall, Paul Clough, and Mark Sanderson. Overview of the TREC 2011 session track. In *TREC'11*.

[167] Evangelos Kanoulas, Ben Carterette, Mark Hall, Paul Clough, and Mark Sanderson. Overview of the TREC 2012 session track. In *TREC'12*.

[168] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Patterns of query reformulation during web searching. *Journal of the American Society for Information Science and Technology*, 60(7):1358–1371, 2009.

[169] Chang Liu, Jacek Gwizdka, Jingjing Liu, Tao Xu, and Nicholas J. Belkin. Analysis and evaluation of query reformulations in different task types. In *ASIST '10*.

[170] Jeff Huang. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM '09*.

[171] Yiqun Liu, Chao Wang, Ke Zhou, Jianyun Nie, Min Zhang, and Shaoping Ma. From skimming to reading: A two-stage examination model for web search. CIKM '14, pages 849–858. ACM, 2014.

[172] Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, and Liyun Ru. How do users describe their information need: Query recommendation based on snippet click model. *Expert Systems with Applications*, 38(11):13847–13856, 2011.

[173] Martin F. Porter. An algorithm for suffix stripping. Program, pages 313–316. 1980.

[174] Laura A. Granka, Thorsten Joachims, and Geri Gay. Eye-tracking analysis of user behavior in WWW search. SIGIR '04, pages 478–479. ACM, 2004.

[175] Lihong Li, Wei Chu, and John Langford. An unbiased, data-driven, offline evaluation method of contextual bandit algorithms. *CoRR*, abs/1003.5, 2010.

[176] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. WSDM '11, pages 297–306. ACM, 2011.

[177] Judea Pearl. Causal inference in statistics: An overview, 2009.

[178] Hui Yang, Marc Sloan, and Jun Wang. *Dynamic Information Retrieval Modeling*. Morgan & Claypool, 2015.

[179] Hui Yang, Marc Sloan, and Jun Wang. Dynamic information retrieval modeling (tutorial). In *WSDM '15*, pages 409–410. ACM Press, 2015.

[180] Marc Sloan and Jun Wang. Dynamic information retrieval: Theoretical framework and application. ICTIR'15. ACM, 2015.

[181] Marc Sloan and Jun Wang. Iterative expectation for multi period information retrieval. In *WSDM Workshop on Web Search Click Data*. WSCD, 2013.

[182] Marc Sloan and Jun Wang. Dynamical information retrieval modelling: a portfolio-armed bandit machine approach (poster). WWW '12, pages 603–604. ACM, 2012.

[183] Shuai Yuan, Ahmad Zainal Az Abidin, Marc Sloan, and Jun Wang. Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users. *CoRR*, abs/1206.1:1–44, 2012.

[184] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002.