Adaptive Domain Modelling for Information Retrieval

M-Dyaa Albakour

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

June 2012

Abstract

Modern search engines employ a number of interactive features to assist users in exploring the document collection and expressing their information needs. Providing these features require knowledge about the document collection in the domain, i.e. a domain model. These models may be difficult to obtain and even if they are available, they may become out of date when the document collection changes or the users start to view the domain differently. In this thesis we propose to use implicit feedback left by users while they interact with search engines to build domain models that evolve over time. These models can adapt to changes in the domain as reflected in the search trend of the user population.

We validate these models in two different IR tasks. The major application is *query recommendation* where previous studies focused on Web search in general or did not consider the temporal aspect of recommendations. Our models address these issues as they are targeted to specific domains such as enterprise search or digital libraries. We furthermore devise an automatic evaluation methodology that allows us to perform extensive evaluation of our adaptive models and observe their performance over time. Using query logs collected from two academic institutions, the evaluation framework assesses the impact of different factors on their performance.

The second application of these models is *query session retrieval*. The query session retrieval problem extends the traditional ad-hoc retrieval by taking into account the previous user interactions with the retrieval system within the same session when answering the query. In this context rather than explicitly providing the user with relevant queries to their information needs, our adaptive models implicitly derive query expansions relevant to the user information needs as identified in the sessions by mapping the session to similar sessions inferred from the models.

Declaration

This thesis is the result of my own work, except where explicit reference is made to the work of others, and has not been submitted for another qualification to this or any other university. This thesis does not exceed the word limit for the respective Degree Committee.

M-Dyaa Albakour

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Udo Kruschwitz who made the PhD journey an exceptionally pleasant and enjoyable one. His guidance and tremendous support made it easy to tackle all the challenges I faced throughout that journey.

The research conducted to produce this thesis was part of the AutoAdapt project which was funded by EPSRC (Grant No. EP/F035357/1 and EP/F035705/1). I wish to thank everyone who was involved in the project, especially Dr. Nikolaos Nanas who provided very valuable input and offered his adaptive information filtering platform to run some of the experiments.

Also I need thank all my friends in Essex and London who made me feel like I am at home. However, I will not list their names to avoid forgetting anyone.

Finally, I would like to express my love and gratitude to my family here and over there in Syria for their continuous support and encouragement to pursue my studies and reach the highest goals.

Contents

List of figures

List of tables

Chapter 1.

Introduction

The revolution in Information Technology, marked by the arrival of the World Wide Web, has created media that allow businesses, governments and individuals to create and disseminate information anywhere. The increasing growth of information was accompanied by the emergence of Web search engines such as Google which facilitate rapid and effective access to the information people seek on the Web. Today they become part of our everyday activities as we use them to access information for business, research, leisure, online shopping, etc. According to ComScore[1], the number of queries submitted to Web search engines around the globe in December 2009 was about 131 billion queries.

Web search algorithms have matured over the past years and become reliable so that a query submitted to Google for example typically returns excellent matches. However, this means that people expect comparable experience with search tools in other environments such as intranets, digital libraries, email systems which contain far fewer documents and where the search algorithms have not necessarily been as effective as on the Web (Hawking, 2011).

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

Moreover, although Web search algorithms are effective, users may still formulate ill-defined queries which need to be refined. To tackle that, Web search engines employ a number of interactive features that help users formulate their queries. For example, Google offers suggestions for automatic completion as the user types in a query in the search box. In fact, assisting users during the search process with interactive features is a vital element of all major search engines nowadays. Another example is faceted search employed extensively by online retailers, such as Amazon, which allows users to explore and easily browse the search results.

It was suggested that assisting users in this manner is particularly important for retrieval systems that serve specific collections like an intranet (Mukherjee and Mao, 2004). Building these features requires knowledge about the the document collection, we call it a *domain model*. A domain model in this case could be a taxonomy of all topics covered by the documents, e.g. the *Universal Decimal Classification*[2] which has been used for decades in libraries to classify books. The problem with these models is that they are not necessarily available for some domains and also they are not flexible to changes in the domain and they can easily become out of date. Another example are the emerging collaborative tagging systems commonly known as 'folksonomies' (e.g. Wikipedia categories). These models have the advantage that they are automatically updated and reflect how the users view the collection. However, they are not generally available for specific collections, like intranets or digital libraries.

A less semantically rich form of a domain model than a taxonomy may be automatically extracted from the textual documents. Examples are subsumption hierarchies in (Sanderson and Croft, 1999) or concept association graphs in (Kruschwitz, 2005). Again, the problem here is that these models become out of date if the document collection changes and even if they are updated continuously they do not reflect how the users view the collection and their interests which change over time.

[2] `http://www.udcc.org`

In this thesis, we aim to use search logs of past user interactions to build domain models automatically. Search logs provide a snapshot of the search behaviour of a user population and are considered a rich source of knowledge that can be employed for a variety of applications which make them a valuable asset for any organisation (Maarek, 2012). A very simple example of the value of search logs is that the top 10 searches submitted to the universal search engine Google over a year can highlight the most important global products, events, and popular celebrities of the year. In 2011, 'ipad2', 'steve jobs', and 'adele' were among the top 10 searches in Google worldwide[3] which enlightens us about the most popular events, products and celebrities of that particular year. Another example of a practical application of search logs is deriving spelling correction for user queries (Ahmad and Kondrak, 2005). In fact, the automatic extraction of implicit feedback left by users in search logs has become the centre of attention of much research (Clark et al., 2012).

We are interested in using the search logs as a source of knowledge to build *adaptive* domain models which can be employed in Information Retrieval (IR). These models will be capable of automatically capturing the user population's search trend at a specific point of time. This would be particularly useful for specific domains like intranets where these models can be exploited for recommendation, navigation and contextualisation.

In this chapter, we first motivate our work by giving examples from the search engine of our university's Web site. Then we outline our research questions and list our main contributions.

1.1. Introductory Examples

This section will provide motivating examples that illustrate why we are interested in acquiring adaptive domain models, how they can be built and how they can be used for information retrieval.

[3] http://www.googlezeitgeist.com/en

The reference system that we apply our methods and techniques to throughout the thesis is a search engine that has been running on the Web site of the University of Essex[4] for more than four years (Kruschwitz et al., 2008). It represents a typical academic context and is an example of an intranet search engine we are targeting in this thesis. The search engine employs an element of interactivity similar to query recommendations proposed by state-of-the-art Web search engines. It responds to a user query by presenting the top matching documents from a backend open source search engine along with some suggestions for query modification i.e. query recommendations. The output looks like that shown in Figure 1.1 (a sample screenshot of the system following the frequent user query *"timetable"*).

We know that queries submitted to search engines are typically very short, normally between two and three words long (Silverstein et al., 1999; Spink et al., 2001), and the majority of queries result in a large set of matching documents even in small collections, e.g. (Kruschwitz, 2003). We assume that a large proportion of user queries can be answered by inspecting the documents returned by a state-of-the-art search engine. However, we also observe that there is a percentage of queries that cannot be answered with a one-shot query because they are ambiguous or very generic. The domain models we build in the context of this thesis can be used to assist searchers who submit these queries by providing more accurate query recommendations.

Figure 1.1 illustrates that an initial query is not necessarily answered by a set of matching documents, but the searcher is offered suggestions for query reformulation. The system derives these suggestions from both an automatically acquired domain model and the best matching documents. Specifically, the domain model is a term association graph. It is constructed in an automated offline process and exploits the markup structure found in the documents of the entire collection and is explained in more detail in (Kruschwitz, 2005).

[4] `http://search.essex.ac.uk`

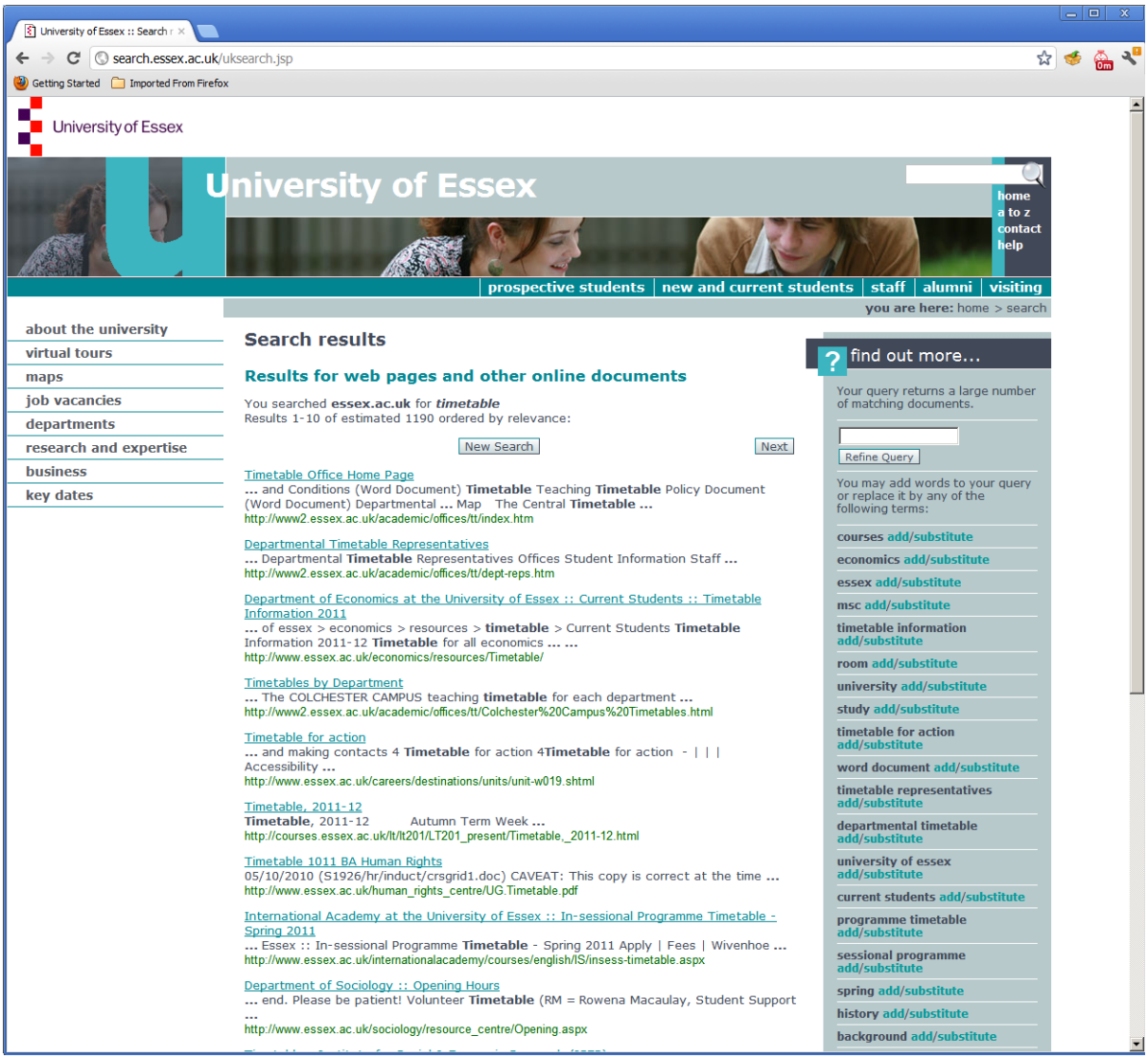


Figure 1.1.: System response to user query *"timetable"*

The problem with these recommendations is first that they may become out of date when the document collection changes, which happens all the time in a University environment. For example, some courses are updated each year. Also users' interests may change over time. This may be due to new events happening in the environment or seasonal events (e.g. the exam period or the summer schools). By seasonal in this case, we mean events that happen during a certain period of the academic year. The adaptive models try to address these issues in query recommendation. In this section, we provide an example of an adaptive model built from the

search logs of our reference system and outline how it can be used in query recommendation and other IR tasks.

1.1.1. Building an Adaptive Domain Model from Search Logs

Figure 1.2 represents a *part* of a domain model of the University of Essex built entirely from recorded users' interactions with the search engine of the university's Website over a certain period of time. This particular model takes the shape of a directed graph where the nodes represent the queries and the weighted directed edges encode an association between the queries. An edge originating from the node 'timetable' to the node 'exam timetable' means that 'exam timetable' is a good candidate for refining an initial query 'timetable' and the weight on the edge is representing the strength of the association. The structure of the graph and the weights on the edges are related to a certain point of time and are continuously updated. For example imagine a situation where a new bus service called 'X22' to Stansted airport was introduced, a new association may be inferred and a new edge from the query 'timetable' to a query 'X22 timetable' is created. Also the weights on the edges may change to reflect changes in the document collection or in the user population's behaviour. The weight 0.4 on the edge from 'timetable' to 'exam timetable' may be reduced after the exam period has passed and the weight on the edge to 'courses' may increase at the beginning of the academic year when students start having classes.

In this thesis, we consider building adaptive domain models similar to the directed graph in the example, i.e. models which only associate terms or queries without encoding any semantics which are *structurally* similar for example to concept association graphs in (Kruschwitz, 2005) or the subsumption hierarchies in (Sanderson and Croft, 1999). The difference is that they can be continuously updated over time and they reflect the user interests as derived from the search behaviour.

How can we build this model from search logs?

Our approach is to continuously observe user interactions with the search engine (the implicit user feedback) to infer structures similar to the previous one. Here we discuss some ideas which we will investigate in the thesis to build a model similar to the previous example.

To build a graph similar to the previous example, we can identify search sessions of individual users and then for each session identify the sequence of queries submitted, we call these query flows. Reformulation of queries can be then easily interpreted to add nodes or edges to the graph or to adjust the weights on the edges over time. Let us assume that we start with an empty model i.e. no nodes or edges exist. An occurrence of a reformulation from 'timetable' to 'exam timetable' will create nodes for each query and an edge originating from 'timetable' to 'exam timetable'. The edge may not be added until a certain number (a threshold) of occurrences of the reformulation is observed. More occurrences of the reformulation will result in the increase of the weight on the edge of the nodes representing those queries. Also over time, some queries or reformulations may become unpopular (not observed) so their weights can be reduced. Imagine that the exam period has passed and the reformulation from 'timetable' to 'exam timetable' is no longer observed, this could be interpreted as a decrease on the edge weight.

We may also observe the user interactions after submitting each query and infer the outcome of the query in terms of how satisfied the user with the results to decide how strong the association should be. This could be done by inspecting the clickthrough data. We can look at the number of clicks as an intuitive indication of the outcome of the query. We can also employ a more elaborate approach where we use the clicks as relevance judgements and estimate the retrieval metrics for the query to help us adjust the weights on the edges.

Note that this process of adaptive domain modelling can be seen as building a profile for a community of users as opposed to building a profile for an individual user.

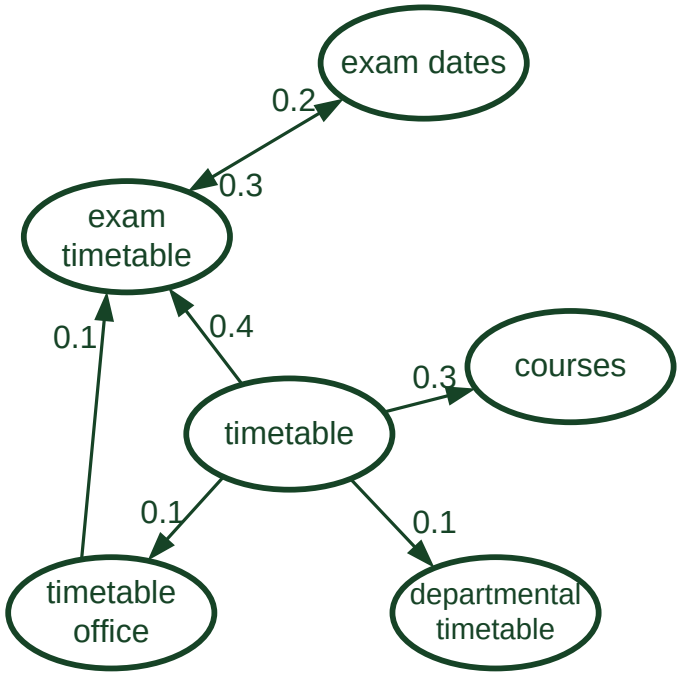


Figure 1.2.: A partial domain model learnt from query logs

1.1.2. Applying the Adaptive Model for IR

Let us illustrate how the adaptive domain model can be employed to serve IR tasks using the previous example

The domain model in the previous example can be used to *recommend queries* by starting from the initial query node and then traversing the graph edges to identify and rank associated query nodes. For the query ‘timetable’ and using the graph one can for example recommend all the directly associated nodes as query recommendations and rank those using the edge weights which would result in the list ‘exam timetable’,‘course’,‘timetable office’, ‘departmental timetable’.

The list of recommendations or the graph itself can be seen as an *interactive feature* of the interface to aid a user in exploring the document collection, the domain and the possible routes of their searches. If the query ‘timetable’ was submitted to our reference system and the

graph was displayed to the user, it will quickly inform the user about the domain. By a quick examination of the graph someone can for example identify that it is a university domain and that there is an entity in the university called 'timetable office' possibly looking after scheduling classes.

Also the graph can be used as a method for *query expansion* which is a common technique in IR where terms related to a query are added to the original query to improve the retrieval performance (Baeza-Yates and Ribeiro-Neto, 2011). We can consider the list of recommended queries as a candidate list for query expansion.

Finally, we also envisage using the adaptive models for enhancing *retrieval over a user session*. In this context the retrieval system does not respond to an ad-hoc query but it rather considers the past user interactions in the search session before the query in question. The adaptive models can be used to derive expansions relevant not only to a query but also to the entire user search session. This can be done for example by using an intersection of recommendations for all queries in the session.

In this thesis, we introduce a range of adaptive domain modelling approaches and evaluate our models extensively for query recommendation and the session retrieval problem.

1.2. Research Questions

The primary aim of the research conducted is to build adaptive domain models and apply them for IR tasks. In particular, the main research questions that we attempt to answer in this thesis are as follows:

1. *Can query logs be used as as a source of implicit feedback to build and adapt domain models that reflect the search behaviour of the user population in a given domain at a certain point of time?*

To answer this question, we investigate two adaptive algorithms, which are biologically inspired, that can effectively digest query logs to build and adapt domain models.

2. *Can these adaptive domain models be applied for IR tasks?*

   To answer this question, we thoroughly evaluate the adaptive domain models mainly for query recommendation but also we consider another challenging IR problem, which is the query session retrieval.

3. *Can clickthrough data be interpreted such that it is used as an implicit feedback to enhance the quality of the adaptive domain models?*

   To answer this question, we investigate a number of approaches for both interpreting and encoding the clickthrough information in our adaptive domain models.

1.3. Thesis Contribution

The statement of the thesis is that implicit user feedback can be used to continuously update a domain model that can be useful for a number of IR tasks. In particular, these models are useful in specific collections where Web search algorithms may perform poorly and where knowledge sources about the domain is difficult to obtain.

The thesis presents two adaptive domain modelling approaches which are biologically inspired and use query flows as source of implicit feedback knowledge. More specifically, the models are based on ant colony optimisation and Nootropia, an immune inspired Information Filtering system. We also extend a state-of-the-art domain model that uses query flows to incorporate click through data.

To evaluate these models for query recommendation, we devise an automatic evaluation methodology that exploits the query logs to perform reproducible and extensive experiments

that assess the quality of query recommendation over time. The evaluation methodology allows us to draw conclusions about these models and the evaluation methodology itself is a contribution which can foster rapid development of adaptive query recommendation systems.

Moreover the thesis contributes to addressing the session retrieval problem by applying the adaptive models to enhance retrieval over query sessions. Some of the introduced models outperform a number of alternatives.

1.4. Structure of the Thesis

The thesis is structured as follows:

- In Chapter 2 we review the related work in some areas of IR which are related to the topics covered in this thesis as IR is a very broad area. We also review the area of domain modelling in various disciplines highlighting the differences in approaches and interests in those communities.

- We split the domain modelling approaches in two chapters. Chapter 3 presents the biologically inspired model to build adaptive models from query flows. Chapter 4 looks at incorporating click-through data in the adaptive domain models. In particular we extend the query flow graph which is a state-of-the-art model for query recommendation and then we propose a more elaborate technique to incorporate click information.

- We devise an evaluation methodology in Chapter 5 where we validate the methodology with a user study. The methodology is then exploited in Chapter 6 to evaluate the models introduced in Chapters 3 and 4 for query recommendation.

- In Chapter 7 we show how a number of adaptive models can be used for the session retrieval problem and evaluate our methods using the TREC session track.

- Finally we give the reader a summary of the work and plans for future work in Chapter 8.

1.5. Publications

The materials introduced in this thesis contain work published in earlier conference papers or journal articles, namely (Adeyanju et al., 2012a,b; Albakour and Kruschwitz, 2011; Albakour et al., 2011a,b,c,d,e, 2012a,b,c; Clark et al., 2012; Kruschwitz et al., 2011):

- I. Adeyanju, D. Song, M-D. Albakour, U. Kruschwitz, A. De Roeck, and M. Fasli. Adaptation of the Concept Hierarchy model with search logs for query recommendation on Intranets. In Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012), 2012a.

- I. Adeyanju, D. Song., F. Maria Nardini, M-D. Albakour, and U. Kruschwitz. RGU-ISTI-Essex at the TREC 2011 Session Track. In Proceedings of the 20th Text Retrieval Conference (TREC 2011). NIST, 2012b.

- M-D. Albakour and U. Kruschwitz. University of Essex at LogCLEF 2011: Studying Query Refinement. In CLEF (Notebook Papers/Labs/Workshop), 2011.

- M-D. Albakour, U. Kruschwitz, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Enriching Query Flow Graphs with Click Information. In Proceedings of the 7th Asia Information Retrieval Societies Conference (AIRS 2011), pages 193 - 204, 2011a.

- M-D. Albakour, U. Kruschwitz, N. Nanas, D. Song, M. Fasli, and A. De Roeck. Exploring Ant Colony Optimisation for Adaptive Interactive Search. In Proceedings of the 3rd International Conference on the Theory of Information Retrieval (ICTIR 2011), Lecture Notes in Computer Science, pages 213 - 224, Bertinoro, 2011b. Springer.

- M-D. Albakour, U. Kruschwitz, J. Niu, and M. Fasli. University of Essex at the TREC 2010 Session Track. In Proceedings of the 19th Text Retrieval Conference (TREC 2010). NIST, 2011c.

- M-D. Albakour, D. Lungley, and U. Kruschwitz. An adaptive search framework for intranets. In Report of the symposium "Lernen, Wissen, Adaptivitäv" of the GI special interest groups KDML, IR and WM (LWA2011), pages 219 - 220, 2011d.

- M-D. Albakour, N. Nanas, U. Kruschwitz, M. Fasli, Y. Kim, D. Song, and A. De Roeck. AutoEval: An Evaluation Methodology for Evaluating Query Suggestions Using Query Logs. In Proceedings of the 33rd European Conference on Information Retrieval (ECIR 2011), volume 6611 of Lecture Notes in Computer Science, pages 605 - 610, Dublin, 2011e. Springer.

- M-D. Albakour, U. Kruschwitz, N. Nanas, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Analysis of Query Reformulations in a Search Engine of a Local Web Site. In Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012), volume 7224 of Lecture Notes in Computer Science, pages 517 - 521. Springer, 2012a.

- M-D. Albakour, U. Kruschwitz, B. Neville, D. Lungely, M. Fasli, and N. Nanas. University of Essex at the TREC 2011 Session Track. In Proceedings of the 20th Text Retrieval Conference (TREC 2011). NIST, 2012b.

- M-D. Albakour, F. Maria Nardini, I. Adeyanju, and U. Kruschwitz. Studying Search Shortcuts in a Query Log to Improve Retrieval Over Query Sessions. In Proceedings of the ECIR'12 workshop on Information Retrieval over Query Sessions (SIR 2012), 2012c.

- M. Clark, Y. Kim, U. Kruschwitz, D. Song, M-D. Albakour, S. Dignum, U. Cervino Beresi, M. Fasli, and A. De Roeck. Automatically Structuring Domain Knowledge from Text: an Overview of Current Research. Information Processing and Management, 48(3): 552 - 568, 2012.

- U. Kruschwitz, M-D. Albakour, J. Niu, J. Leveling, N. Nanas, Y. Kim, D. Song, M. Fasli, and A. De Roeck. Moving towards Adaptive Search in Digital Libraries. In Advanced Language Technologies for Digital Libraries, volume 6699 of Lecture Notes in Computer Science, pages 41 - 60. Springer, 2011.

Chapter 2.

Background

In this chapter we give a comprehensive discussion of the related work in the literature.

As we are proposing to build some adaptive domain knowledge for search, we break down the review into two parts. First we look at IR in general and review those areas in IR which are related to the thesis contributions (Sections 2.1 - 2.6). Then we discuss the topic of domain modelling in Section 2.7.

In the first part (Sections 2.1 - 2.6), our aim is to review the key IR areas where the materials presented in this thesis can add value to. In particular, we will look at interactive IR and outline current directions in this area to position our work. Then we discuss query recommendation techniques and outline the challenges and gaps in this area that still need to be addressed. Also we review the areas of query log analysis and implicit user feedback to highlight the lessons learnt there and take them forward when we build our adaptive models. In addition we give the reader a background of enterprise IR challenges and outline its differences with Web IR in general.

In the second part (Section 2.7), we will look at the topic of domain modelling which has attracted quite a lot of interest in a wide range of communities with different applications in mind.

2.1. Information Retrieval

Information Retrieval is a broad area that finds its roots in a number of disciplines. This includes digital libraries, information sciences and computer science. Information retrieval fundamentally deals with providing users access to information they seek (Baeza-Yates and Ribeiro-Neto, 2011). In a classical IR system, the IR process can be described as follows. The user expresses her information needs as a *query* and the system *ranks* the documents it maintains according to how much they satisfy the user information needs. Documents which are believed by the system to be more relevant than others are presented to the user for inspection and the user can iteratively apply the querying process depending on how much information they gained.

Early developments in IR research dates back 50 years ago when the Cranfield experiments were published (Cleverdon, 1962, 1967) which helped to define an evaluation framework for IR ranking models. After that a number of retrieval ranking models (models where a ranking function that can score textual documents for a given textual query) were developed such as estimating similarities based on the vector space model (Jones, 1972), probabilistic models (Robertson, 1977) and language models (Croft and Lewis, 1987).

With the emergence of the World Wide Web as the largest medium to disseminate information, information retrieval has grown from a small discipline in information sciences to an everyday experience for billions of people around the globe (Rao, 2004). Web IR relies on both the textual content of documents and the unique link structure of the Web. The Web search engine Google originally introduced the popular PageRank algorithm (Brin and Page,

1998) which exploits the link structure in the Web. Since then, Google has grown in fame and currently possesses 81% share of the Web search market[1].

The emergence of the Web and the growth of Web search has resulted in an increasing interest in IR research. In addition to that, the Text REtreival Conference (TREC) played a great role in fostering IR research in the last decades by creating large test collections (Vorhees and Harman, 2005). Today IR research has a wide spectrum of branches and applications. This includes indexing techniques, Web crawling, multimedia retrieval, interactive retrieval, relevance feedback among others.

In the rest of the chapter, we will review some of the IR sub-areas that directly or indirectly relate to our contributions. We will highlight the advances and challenges in those areas to position our work with regards to the state-of-the-art techniques.

2.2. Interactive Information Retrieval

Interactive Information Retrieval (IIR) deals with how people use and interact with an IR system to find the information they seek (Ruthven, 2008). The Cranfield framework (Cleverdon, 1962) which dominated IR evaluation simplifies IR research by abstracting the user interaction with a one-shot query representing the user's information needs. However real-world searches may contain query reformulations, result examination and browsing which are not captured in the classical IR ranking models. Recently Belkin (2008) has called the move beyond the limited, inherently non-interactive models of IR to truly interactive systems the *challenge of all challenges* in IR at the moment.

IIR has received much attention in recent years, e.g. (Marchionini, 2008; Ruthven, 2008; Tunkelang, 2009). Furthermore, increased activity in developing interactive features in search systems used across existing popular Web search engines suggests that interactive systems

[1] `http://www.netmarketshare.com` (May 2012)

are being recognised as a promising next step in assisting information seeking. For example, prominent Web search engines have added more and more interactive features, including both suggestions as the user types as well as related searches and modification terms following the query submission (e.g. Google, Yahoo! and Bing). There is also evidence that integrated interactive IR systems can offer significant advantages over baseline systems (Yuan and Belkin, 2007). The Interactive aspect of IR systems has also been recognised as an important issue with the introduction of the interactive track in TREC (Vorhees and Harman, 2005). More recently, the Session Track was introduced in TREC 2010 (Kanoulas et al., 2011, 2012). It aims at the analysis of user interactions in a search session to improve the retrieval performance throughout the session.

Here we review some of the methods and features that have been found effective in *assisting* users when developing IR systems.

One of the main issues that users face in interacting with an IR system is the ability of *formulating* the right query that best represents their information needs. This is even more challenging in *exploratory* search, where a user's existing knowledge about the topic sought is limited (White and Roth, 2009). Supporting a user in the search process by *interactive query suggestions* is one avenue to address this issue and it has been discussed extensively (Efthimiadis, 1996). Query suggestions can help in the search process even if they are not clicked on (Joho et al., 2004; Kelly et al., 2009). Studies have shown that users want to be assisted in this manner by proposing keywords (White and Ruthven, 2006), and despite the risk of offering wrong suggestions they would prefer having them rather than not (White et al., 2007). In Section 2.4, we give the reader a more detailed discussion about the area of query recommendation. Another promising interactive search feature is the exploitation of meta-data to provide users with the ability of browsing and filtering documents with faceted search (Ben-Yitzhak et al., 2008). Faceted search is employed extensively by online retailers, such as Amazon[2]. Offering effective visualisation of information that provides insight for the users

[2] `http://www.amazon.com`

of the search results and document collection makes search more effective, e.g. the Relation Browser in (Marchionini and Brunk, 2003) and AquaBrowser[3] which is a navigational tool applied in digital libraries.

The emergence of social networks such as Facebook and Twitter has triggered the research of collaborative information seeking systems which can offer diverse perspectives, experience, and expertise to the search process (Golovchinsky et al., 2009; Smyth et al., 2009).

As a summary, the interactive aspect of IR systems is an important issue. Nowadays IR systems are investing more in interactive features to support the users during their information seeking process. Our work aims at contributing to the IIR research by building adaptive domain models which can either support users by explicitly assisting them with related search terminology or allow them to navigate through the information space.

2.3. Query Log and User Behaviour Analysis

Query log analysis aims to capture the users' behaviour in interacting with a search system reasonably and in a non-intrusive manner. It provides insight about how users formulate their queries (e.g. the average number of terms in a query), the characteristics of search sessions (e.g the average length of a search session, the average number of queries in a session), the browsing behaviour (e.g. the number of results pages viewed, the number of clicks, etc.), the topics searched and the trend of queries or topics over time (Spink and Jansen, 2004). This would have an impact on designing more effective search engines. In fact, it is recognized that there is a great potential in mining information from query log files in order to improve a search engine (Jansen et al., 2008; Silvestri, 2010).

Here we provide the reader with some examples of previous research findings on user search behaviour using log analysis.

[3] `http://serialssolutions.com/aquabrowser/`

Early work on Web search log analysis included (Silverstein et al., 1999) where they analysed a query log consisting of nearly a billion queries submitted to the AltaVista search engine in a period of 6 weeks in 1998. They found that queries on average are rather short and contain between 2 and 3 terms and that only a small portion of queries appear more than 3 times. In 85 % of the cases users viewed more that one page of results. They also found that in about 22% of the sessions users had to reformulate their queries with around 2 queries on average per user session. No topic-based analysis was reported in this study. Later findings in (Jansen et al., 2000; Spink et al., 2001) were consistent with these statistics, but Spink et al. (2001) shed some light on the topics searched by randomly sampling the queries and manually assigning a query to a category. Spink and Jansen (2004) provided a comprehensive survey of log analysis and presented findings on three different Web query logs: AllTheWeb.com, AltaVista and Excite. They reported changes in the above statistics over the years where the average number of terms per query was increasing in the studied logs and that search sessions with only one query declined. They also reported a dependency between a Web search engine and the distribution of the topics searched.

Later studies considered examining the trend of topics over time. Beitzel et al. (2004, 2007) proposed a framework to detect topical trends over time. Using AOL logs they found that certain categories (topics) can become popular over a few hours in a day (short-term trends) or over several weeks or months (long-term trends). Similarly Wang et al. (2003) analysed search logs of a university search engine and reported trends of queries during a certain season, month, or a day.

In this thesis we take this observation forward and attempt to build adaptive domain models which can reflect these trends in user search behaviour.

Another interesting aspect of user behaviour analysis that can be inferred from the logs is query reformulation analysis. One of our aims in this thesis is to develop methods that assist users in query reformulations and therefore it is important to understand how people

reformulate their queries. Previous studies on query reformulations include the classification of query reformulations found in search logs (Anick, 2003). In this work, reformulations can be classified into different categories. The main categories are generalisation, specification and parallel moves which map to the three possible navigational moves through a hierarchy of topics. Other studies include analysing the effectiveness of the query reformulation types in different search tasks (Liu et al., 2010). Huang and Efthimiadis (2009) evaluated the effectiveness of query reformulation strategies by observing the clickthrough information before and after the reformulation.

2.4. Query Recommendation

When people use Web search engines, they often formulate their information needs as a textual query and submit it in the search box to get back a list of documents. Studies on Web search engines found that these queries are often short (2-3 words on average (Jansen et al., 2000)) and therefore they do not adequately satisfy users' information needs. After evaluating the results of their initial queries, users may modify or reformulate their queries until their needs are satisfied. Query reformulations have been found to be very common in search logs (Jansen et al., 2007). Empirical studies have found that the average length of a query session on Web search engine is 2-3 queries (Silverstein et al., 1999; Spink and Jansen, 2004). Assisting the users to identify the right queries for their information needs is therefore important. In fact it was also stated that searchers may have difficulties in identifying useful terms for effective query expansion (Ruthven, 2003).

Query recommendation addresses this particular issue. It is the task of explicitly suggesting related queries for a searcher's initial query. It was found to be a promising direction for improving the usability of Web search engines (Joho et al., 2004; White and Ruthven, 2006; White et al., 2007).

Therefore there has been a lot of interest in the IR community to develop and evaluate query recommendation systems.

A query recommender system maintains a domain model and defines a process for extracting suggestions using the maintained domain model. Later in this chapter, we discuss the concept of domain modelling and its objectives in various disciplines. In this section we review the techniques studied to build domain models for query recommendation and utilise them for suggesting queries to the user. We also aim to highlight the challenges associated with building query recommendation systems.

In Figure 2.1 we provide the reader with a guide of reading this section. It illustrates a breakdown of the approaches we found in the literature to develop query recommendation systems. Our classification is based on the resources used for building the domain model to recommend queries.

We identify three main streams for building query recommendation systems:,

1. *Log-based approaches* mine recommendations from large search logs recorded by search engines.

2. *Content-based approaches* rely on the textual content of the document collection searched by the user.

3. *Hybrid methods* use both the search logs and the document collection and other knowledge sources to derive recommendations.

Query recommendation systems can also be seen as a specialised discipline in the broader area of recommender systems (Adomavicius and Tuzhilin, 2005). Recommender systems attracted much research due to their practical applications that help users to deal with the information overload by providing personalised recommendations of goods, services and content online. While generally the focus of recommender systems is to correctly build a profile of preferences for a single user, the query recommendation systems, which is the focus of

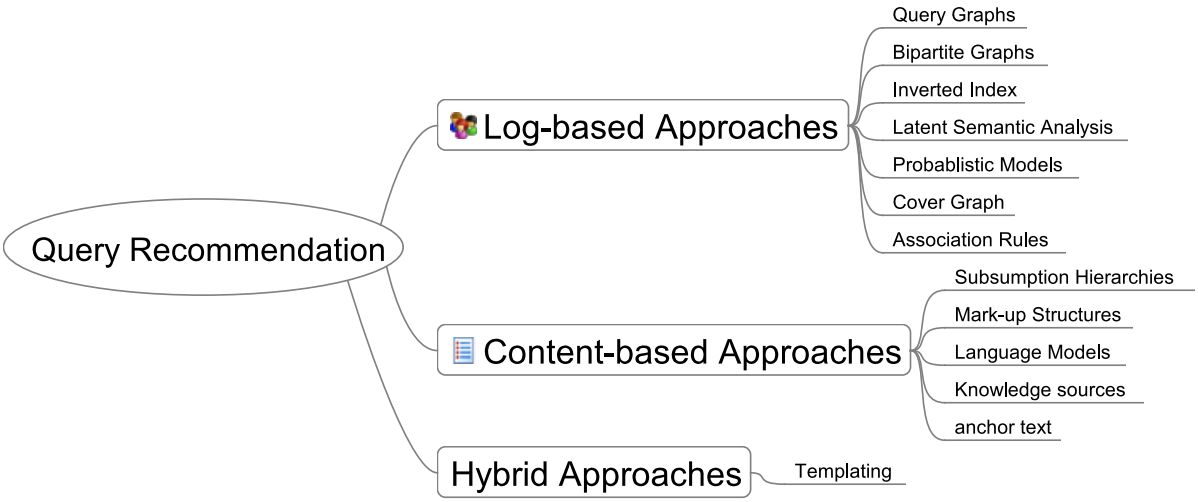


Figure 2.1.: Taxonomy of query recommendation methods

this thesis, mainly aim to provide recommendation that can assist the entire user population, i.e. there is no single user profile but instead a community profile. Approaches to build recommender systems can be classified into (a) collaborative filtering (Hill et al., 1995) where the user is recommended items that people with similar tastes and preferences liked in the past; (b) content-based approaches (Pazzani and Billsus, 1997) where the user is recommended items similar to the ones the user preferred in the past; and (c) hybrid approaches which combines the previous two. This classification is similar to our classification of query recommendation systems. Collaborative approaches are akin to log-based query recommendation. However, the major difference is that the majority of collaborative filtering approaches rely on explicit user judgements (e.g. rating a movie with 5 stars, buying a book), whereas log-based query recommendation systems rely on implicit user feedback embedded in query reformulations and clicks on search results.

2.4.1. Log-based Approaches

Log-based approaches exploit past user interactions with the search engine as recorded in the logs. With the increasing availability of search logs, a lot of methods have been developed for mining logs to capture "collective intelligence" for providing query recommendations.

Here we discuss a wide range from the spectrum of methods used and highlight the main challenges and issues with these methods of which most were developed for Web search engines.

Early work investigated the power of association rules applied on queries in the same session to derive related queries (Fonseca et al., 2003). This simple method was found to be effective for product search on the Web. Developed originally in (Boldi et al., 2008), the query flow graph is a directional graph derived from large query logs where nodes represent queries and an edge between two nodes (q, q') represent the fact that at least one user has reformulated their queries to q' after submitting the query q. The graph can be then used to recommend related queries by performing a random walk starting from the given query. Later work include projecting the query flow graph for diversifying query recommendations (Bordino et al., 2010) Anagnostopoulos et al. (2010) improved the model by optimizing a utility function for the random walk. The function tries to maximize the chance of a termination of a user session or finding a landing page. Jones et al. (2006) combined mining query logs with query similarity measures to derive query substitutions which can be recommended to the user or generate alternative queries for sponsored search.

So far all the aforementioned methods rely on a sequence of queries in a session to recommend related queries. But the implicit feedback left by users during a search session contains other information such as the documents displayed after each query and those clicked with their dwelling times. Hence, clickthrough information can be used for query recommendation systems. An example is the *bipartite query-click graph* in (Craswell and Szummer, 2007),

where queries and documents are represented as two types of nodes in the graph and edges between a query and a document denote a click on that document after issuing that query. Queries pointing to the same document can be considered related for query recommendation (Wen et al., 2001). Mei et al. (2008) developed an algorithm to derive query suggestions from the query-click graph using random walks with hitting time. Ma et al. (2008) used the bipartite graph to derive a query similarity graph with latent semantic indexing. The *cover graph* developed in (Baeza-Yates and Tiberi, 2007) utilises the same concept of the query-click graph. It is a non-directional query graph where an edge between two queries denotes that they share at least one click on the same document. The edges are weighted with the cosine similarity in a vector space where the dimensions represent all the documents and queries are vectors with components weighted with the number of clicks on each document. The cover graph was found effective to derive semantically related queries.

There are a number of issues and shortcomings that were not addressed in the aforementioned techniques. Here we review some of these issues and how they were tackled.

- The problem of query recommendation may be extended to provide query suggestions not only to an ad-hoc query but also to the entire context of the user search session. In other words previous queries submitted by the user within the same search session may be taken into account to provide *context-aware* suggestions. The work of (He et al., 2009) is similar to the query flow graph model but they used a probabilistic model to predict the next user query after a sequence of queries in a session. Query recommendations with random walks on the aforementioned query flow graph model can be modified to provide suggestions using queries in the entire user session (Boldi et al., 2008).

- Although these methods have proven to be successful on some test queries, they often fail to satisfy rare or unseen queries. Search Shortcuts (Broccolo et al., 2012) use an inverted index where sessions are treated as documents and the index is then searched for successful sessions to recommend to the user. Search Shortcuts have shown to be very

effective for queries on the long-tail of the distribution of queries i.e. rare queries or even unseen queries.

- Another issue not very well studied and understood is the effect of ageing over time for query suggestions. Suggestions derived using the previous methods treat the search logs as one batch and do not take the temporal aspect into account. Hence, the recommendations may become out of date and therefore irrelevant. Baraglia et al. (2010) illustrated that a query flow graph model can become outdated over time. Broccolo et al. (2010) implemented an incremental version of the 'association rules' and the 'query flow graph' methods where the model can be updated whenever a new query is submitted.

  In this thesis we will be addressing this particular issue in query recommendation systems. We will look at models which can be adaptive over time and we will assess their performance over time.

- Most of the methods discussed were studied on commercial Web search engines, where the logs are particularly so large, however recently query recommendation for enterprise search has gained some interest, for example (Al Hasan et al., 2011).

- Most studies focused on offering suggestion in the same language. Studies on query recommendations go beyond that and also consider cross lingual recommendation, i.e. given a query in one language, the system recommends queries in other languages (Gao et al., 2010).

2.4.2. Content-based Approaches

The content based approaches rely on the actual contents of the documents or some knowledge sources about the content of the document to derive query recommendation. Early work of the *content-based* approaches to suggest relevant queries includes the subsumption hierarchies (Sanderson and Croft, 1999). In this work, the authors introduced a hierarchical relationship,

imposing a subsumption relation between pairs of extracted concepts by comparing the two sets of documents retrieved for the pairs when submitted to a search engine as queries.

Web document structure as represented by hypertext markup language (HTML) has been exploited in conjunction with frequency analysis. For example, Kruschwitz (2003) used the count of different structural contexts as a guide for extracting concepts from Web pages. Concepts are then associated if they co-occur in the same pages and then used for query recommendation. Another remarkable approach for exploiting the hypertext structure is the use of anchor text in (Kraft and Zien, 2004) for mining query refinements as they call it. With this approach anchor text representing a document is indexed and the query is matched against that index to find related terms to suggest to the user. Language modelling can also be exploited for dynamic query recommendation as proposed by Bhatia et al. (2011).

2.4.3. Hybrid Approaches

Hybrid approaches use both search logs and content or knowledge resources describing the content. Szpektor et al. (2011) and Bortnikov et al. (2012) used a query flow graph model but introduced a templating approach for queries when building the graph or suggesting queries with the graph. With templates a query like 'jaguar transmission fluid' will be mapped to a '<car> transmission fluid', and a recommendation like 'Jaguar used parts' can be constructed from a template '<car> used parts' found in the templates graph. The templates are created using external knowledge taxonomies, namely Wordnet 3.0 hypernmy hierarchies, Wikipedia categories and YAGO[4].

[4] http://mpi-inf.mpg.de/yago-naga/yago

2.4.4. Other Aspects of Query Recommendation

We have already discussed a wide range of techniques for query recommendation and outlined their advantages and disadvantages. Our discussion was simplistic is the sense that we focus on the task of finding related queries for a given query. However implementing a query recommender system imposes more challenges.

For example, the *visualisation* issue of query recommendations and their *usability* is an important issue. Researchers investigated more elaborate techniques than plain lists to offer suggestions to the user. Joho et al. (2002) studied hierarchical menus to visualise suggestions and they found that the users can finish their search tasks more quickly using these interfaces when compared to plains lists ordered alphabetically. Kato et al. (2012) studied the idea of clustering recommendations on the user interface and found out that users consider suggestions presented in this manner more helpful and that they can be more successful in their search tasks when using these interfaces

Another form of query recommendation which has become popular in commercial Web search engines such as Google is query auto-completion, which suggests queries to the user as the user types the query in the search box. Kamvar and Baluja (2008) studied these interfaces on mobile phones and found out that users prefer to have autocompletion features rather than not and that these interfaces save them time and effort.

2.4.5. Evaluation of Query Recommendation Systems

We identify three main streams of approaches for evaluating the quality of query suggestions. In most cases, these techniques are combined and unlike commonly understood TREC measures for retrieval quality (such as Mean Average Precision) there is no commonly agreed measure for query recommendation.

1. *User studies*: With this evaluation the users are consulted in different ways. One approach is to derive recommendations from the system using a sample of queries and then ask the users to provide relatedness or usefulness judgements (Boldi et al., 2009; Kraft and Zien, 2004; Sanderson and Croft, 1999). Another way of consulting users is to perform a task-based evaluation such as those in (Kelly et al., 2009). In this case, the users conduct an information retrieval task using the recommender system. Users can then judge whether these recommendations helped them in achieving their tasks (Bhatia et al., 2011; Kato et al., 2012; Kruschwitz, 2003).

2. *Using external knowledge sources as gold standard*: With this evaluation the recommendation provided by the system are matched against an external semantic knowledge taxonomy. For example, Baeza-Yates and Tiberi (2007) used the Open Directory Project (ODC) categories to examine if a suggested query by the recommender system fills in the same category of the query in question.

3. *Evidence from query logs*: This is particularly used with log-based approaches. Under this evaluation the log data is divided into training and testing data. The training data is used to learn the model and the testing data provides the ground truth. Usually query reformulations observed in the query log are used as 'gold standard' judgements (He et al., 2009; Szpektor et al., 2011).

Each of these approaches has its own advantages and limitations. Users studies can accurately measure the improvements on the end user, but they can be expensive. Evaluation with external knowledge resources requires the knowledge to be available which is not necessary the case for local domains such as the ones we consider in this thesis. Using query logs is quick and cheap and allows for thorough evaluation, but there is no explicit feedback from the end user and qualitative explanations. In this thesis, we aim to focus on using logs as a primary resource to evaluate our adaptive domain models.

2.5. Enterprise Search

Enterprise Search deals with providing information access to proprietary information found in enterprises such as corporations, governments, non-profit or academic organisations. Hawking (2004) defines enterprise IR to include search of an organisation's external Web site, search of an organisation's intranet, and search of other electronic text held by an organisation which can be found in many different forms including emails, proprietary database records, documents, etc.

Developing an efficient and effective search engine for a modern enterprise can have a significant financial value. Feldman and Sherman (2001) highlight the importance of information access in the enterprise. They found that access to information is critical to their activities. They also estimated the costs for an enterprise of not finding information to be in the region of $2.5 to $3.5 millions per year.

The characteristics of enterprise search impose a lot of challenges on IR systems and make it different from Web search (Broder and Ciccolo, 2004; Hawking, 2004; Sherman, 2008; White, 2007). Enterprsie Search is different from Web search in many ways. The size of the Web is huge with billions of documents and users all over the world, estimated as much as 2.2 billion users in 2011[5]. Conversely, an enterprise is likely to contain far fewer documents and has a limited audience which is the staff or the customers of the enterprise. The search tasks are also different in nature as they are motivated by work problems. Hawking (2011) provided examples of tasks which can be supported by enterprise search tools. These include approving an employee travel request, responding to a call in a call centre or finding people with technical expertise in a certain topic. Therefore the notion of a 'good' answer to a query is different between Web search and enterprise search. In the former a 'good' answer would typically return as many relevant documents as possible because the number of relevant documents can be very large. In the enterprise a 'good' answer is often the 'right' answer and there is

[5] `internetworldstats.com`

often one document that contains the answer (Fagin et al., 2003). Moreover, content in the enterprise comes from different repositories in different formats (emails, Web pages, database records) and do not necessarily cross-reference each other with hyperlinks as in the Web. For example (Fagin et al., 2003) found out that on the IBM corporate's intranet the ratio of strongly connected components (the ones that reach other by following links) accounts for only 10% of the intranet collection which is much smaller than on the Web. Therefore the popular PageRank (Brin and Page, 1998) algorithm for Web search may not be as effective on intranets. For example, Hawking et al. (2004) found out links on the Web from outside an organisation has little value to add on the quality of the Web site search for that organisation. Although Upstill et al. (2003) showed that variants of the PageRank and anchor text can be effective for Web site Search, simple in-link counts (links within the Web site) achieved most of the benefit of the PageRank.

Developing efficient and effective tools for enterprise search has therefore attracted researchers in the academic and industrial IR communities alike. Fagin et al. (2003) developed an aggregating ranking function that combines documents from different sources and use simple heuristics to effectively aggregate the ranking of these documents. The enterprise track was introduced in TREC to study search tasks such as expert finding and email search (Bailey et al., 2007; Craswell et al., 2005; Soboroff et al., 2006). Mukherjee and Mao (2004) outlined the key ingredients that should be found in an enterprise search system. In addition to various techniques of designing effective ranking models for enterprise search, they suggested that contextualisation and recommendation can be particularly useful for search on the enterprise.

In this thesis, we will take these observations forward and we propose adaptive domain models for local Web sites and intranets to improve retrieval in these environments. These models will be able to capture context and assist users in finding the right terminology that may deliver the 'right' answer to their information needs.

2.6. Implicit User Feedback in Information Retrieval

Traditionally IR research relies on explicit relevance judgments provided by experts for both evaluating IR models which is the evaluation procedure in TREC (Vorhees and Harman, 2005) or for automatically learning or tuning a ranking function, e.g. (Fuhr, 1989). The problem is that relevance judgments are usually expensive and not easily collectable on a large scale. One possible avenue for collecting relevance judgments is to consult the actual users of the retrieval system on providing explicit relevance judgments. However studies show that users are reluctant to leave any explicit feedback when they search a document collection (Dumais et al., 2003; Jansen et al., 2000; Markey, 2007)

Given the reluctance of users to provide explicit feedback on the usefulness of results returned for a search query, the automatic extraction of implicit feedback has become the centre of attention of much research. Implicit feedback left by users has been exploited in a number of applications in IR. We have already discussed how query recommendations can be derived from implicit user feedback found in search logs (cf. Section 2.4).

Here we discuss the related work on what sort of implicit user feedback can be utilised for learning to rank, adapting the ranking function or for evaluation and how it can be used.

Clickthrough data is one form of implicit feedback left by users which can be used to learn a retrieval ranking function. Joachims (2002) showed how a retrieval function can be adapted using support vector machines where pairwise relative preference judgments from click data are used as ground truth for learning. Agichtein et al. (2006) demonstrated how aggregate click statistics can provide significant improvement on the retrieval performance by either adapting the ranking function or re-ranking the original ranked list of results. These methods learn *offline* as a one-off process on past user interactions recorded in the search logs. Recently Hofmann et al. (2011) proposed an *online* learning framework from user clicks where they used reinforcement learning to balance exploration and exploitation. Under this framework, the

system *explores* new solutions (documents) that are presented to the user but at the same time it balances the performance by *exploiting* previously learned satisfactory results.

Implicit user feedback includes other information that goes beyond clicks and researchers have studied the usefulness of such information. Kelly and Belkin (2004) used reading time as implicit feedback but they found out that is difficult to interpret. Fox et al. (2005) developed a Bayesian model that combines 30 different implicit measures including session duration, reading time, scrolling time and the way users exited the search to correctly predict an explicit judgement of relevance. White and Huang (2010) mined user search trails for search result ranking, where the presence of a page on a trail increases its query relevance.

One of the most important challenges in learning from implicit feedback is that user feedback can be noisy, for example Joachims et al. (2005, 2007) found out that clicks should not be treated as absolute judgements and clicks are biased by the order of presentation and the quality of the underlying retrieval model. This work also suggested a number of strategies to generate relative judgements form clicks. For instance, a click on a document can be considered more important than earlier clicks on documents in the same list.

Implicit user feedback was also used by IR researchers for *automatic evaluation* to eliminate the burden of collecting expensive relevance judgements (Soboroff et al., 2001). Joachims (2003) developed a framework for using clickthrough data to evaluate two retrieval functions. The framework is capable of eliciting the order bias effect on clicks. Radlinski et al. (2008) tested two groups of evaluation strategies with clickthrough data: (a) absolute metrics which treat clicks as independent events of the user context and (b) paired comparison tests which uses the previous framework in (Joachims, 2003) to infer relative preferences between two rankings from click data. They found the majority of the absolute metrics are not decisive and the paired comparison tests are capable of identifying a better retrieval function. Similarly, Carterette and Jones (2008) trained a probabilistic click model where they can predict the probability of relevance for each result.

2.7. Domain Modelling

Domain modelling can generally be defined as the process of capturing and structuring knowledge embedded within an information object of interest to a selected domain (for example, a collection, a community, an area of interest). Domain models are realised in many ways, for example, as an organisation of documents into a classification schema, as a linked network of information objects, as a relational database, and as a hierarchical or partially ordered graph comprising domain-relevant entities as nodes. We will focus on the most general formulation of a domain model, described as a selection of concepts or terms judged to be salient within a given collection (whether the collection be a single sentence, an entire library document collection, or an even larger collection) and/or relations between these concepts.

Domain models were rooted initially in *Artificial Intelligence (AI)*, in particular in the field of *Knowledge Representation (KR)* for example, (Quillian, 1967; Woods, 1975). Cyc, the first large-scale knowledge representation project aimed at conceptually capturing *Common Sense Knowledge*, goes back to 1984 and the work on this project is still ongoing, for example, OpenCyc and ResearchCyc (Lenat et al., 1985). Later on, the development and application of domain models further evolved with the emergence of the Semantic Web (SW), and as part of the ongoing research in IR and Natural Language Processing (NLP).

Domain models have been developed in a variety of research disciplines and for various different reasons and as a result numerous (sometimes synonymous) terms have emerged which are all used to refer to the concept of a domain model, such as: *Semantic Network, Ontology, Concept Map, Conceptual Graph, Term Association Graph, Taxonomy*, etc. While these names have been created to convey slightly different notions in the literature, they often overlap in their usages and are employed to refer to an underlying structure, characterised by the general formulation of selected vocabulary and relations between concepts (usually terms) in the vocabulary. Note that the notion of "concept" varies and there is no consensus across

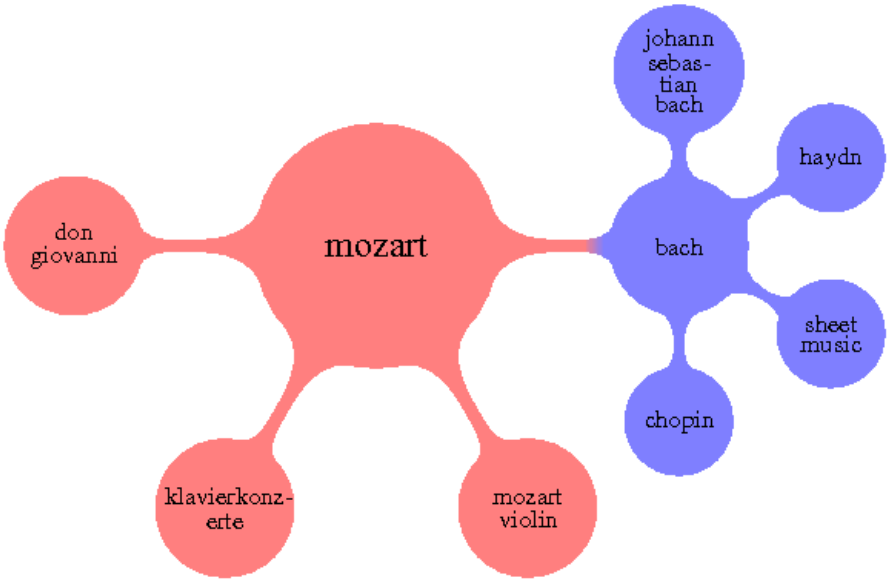


Figure 2.2.: Partial domain model.

different communities. It is beyond the scope of our review to go deep into this issue, and we will not adopt any specific definition of "concept". Instead we will use it in an abstract sense.

Some of the differences and similarities in the various approaches can be illustrated through the example shown in Figure 2.2. This provides a small snapshot of a domain model that has been built from a text collection. As a simple term association graph/network, it shows the links between different terms that are in some way related but the relations between the terms are not formally specified. If, however, the nodes in the model were treated as concepts and the specific types of relations between these concepts were identified, this model could be used to develop *Conceptual Graphs*, a *Semantic Network* or part of an *Ontology*. For example, with reference to the Cyc project, in OpenCyc[6] the concept, "Mozart" is of type *Individual*. This concept includes a number of aliases referring to the same individual ("Wolfgang A. Mozart" and "Wolfgang Amadeus Mozart". This particular concept is of types *classical music performer*, *Austrian*, *composer*, etc. Adding specific interlinking relations, such as the information that

[6]http://openCyc.org

Mozart composed the opera "Don Giovanni" would represent a move towards developing the original simple term association network into a broader, more semantically-enriched structure. A different approach would involve representing only hierarchical relations between terms, for example, the fact that *Mozart* is a composer, a composer is a musician, a musician is an artist, etc. This would result in a different, somewhat simpler knowledge structure. In fact, for this example, these are exactly the relations that can be found in WordNet[7] (Fellbaum, 1998), a large-scale *lexical* knowledge base.

Domain models are, of course, built and used for different purposes. Cyc is an AI project that encodes knowledge which can be used, for example, in automatic reasoning. It is thus very closely related to the idea of the SW which is aimed at bringing *"structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users."* (Berners-Lee et al., 2001). *WordNet*, on the other hand, conceptualises knowledge about the English language which can be applied in NLP, for example, to disambiguate word senses. The partial domain model shown in Figure 2.2 does not derive from either of these two areas but is closely linked to both of them. This model was indeed extracted from the query log files that collect user interactions with a library catalogue search engine. It has been built automatically to suggest query expansion and modification terms in an IR context. In this thesis, we are interested in building this kind of domain models. We will propose approaches that make the automatically-acquired models adaptive, able to improve and change automatically.

Adaptive models are unlike static (traditional AI-style) knowledge sources such as WordNet or Cyc. The advances of automatic construction and adaptation of domain models are addressing the so-called knowledge acquisition bottleneck (KAB), including problems such as acquisition latency, knowledge inaccuracy and maintenance of the acquired knowledge (Cullen and Bryman, 1988; Tang et al., 1994; Wagner, 2006).

[7] http://wordnet.princeton.edu/

To break through the KAB, various research communities have been seeking more effective solutions to the automatic construction and adaptation of domain models. The overall aim of this section is to review the recent development in this direction.

To provide a constructive reference point for this discussion, we have focused on three research streams: (1) artificial intelligence (AI) and semantic web (SW) technology, (2) natural language processing (NLP) and (3) IR. There tends to be surprisingly little overlap between these communities despite the range of shared interests.

It should be pointed out that this categorisation is not intended to be definitive. For example, it could be argued that AI and SW deserve to be treated as two separate areas, whereas in other cases the borders are not so clear-cut. For example, work in information extraction inherits from both NLP and IR.

The simplified categorisation into three fields is intended to help demonstrate the spectrum of characteristics that arise as a consequence of the particular vision within different research areas. These research communities can often be characterised by the types of concepts and relationships between concepts in which they tend to be interested, and this seems to be heavily influenced by the over-arching objectives within each of these communities.

To illustrate the different visions and representations of models in different disciplines, consider the partial models in Figures 2.2 and 2.3. The first model (Figure 2.2) is a simple term association network in which nodes, that is, the model concepts, represent query terms and the relations between these nodes are not defined. As mentioned earlier, these models are very common in the IR community. The second model (Figure 2.3) is part of an ontology in which the nodes refer to entities and the relationships between these entities are semantically defined (*Mozart* composed *Don Giovanni*). In the SW and AI communities such knowledge representation is necessary to allow automatic reasoning and enable Web agents to understand the content on the Web.

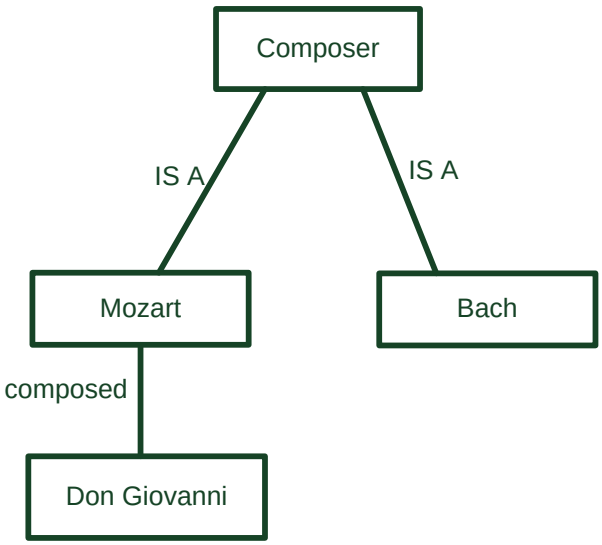


Figure 2.3.: Partial ontology example.

Here we distinguish two main paradigms of building domain models: *data-driven* and *knowledge-driven* approaches. More generally speaking, these could be referred to as statistical and symbolic models.

The data-driven approaches are defined by the emphasis they place on extracting key words or phrases that capture concepts. The relationships included in a data-driven model tend to vary widely in type and granularity reflecting only a loose notion of *relatedness* based on the topic of the text. Some approaches do not attempt to generate relationships at all while others generate relationships between concepts based on degrees of specificity and subsumption. The relationships are often extracted using co-occurrence frequency within the collection or using inferred attributes of the concepts.

The *knowledge-driven* approaches, on the other hand, tend to target specific types of relationships (such as hyponymy, meronymy and synonymy) that are defined a priori to the extraction process. Entities with the corresponding relationships are extracted based on the specified types. For this purpose, lexical databases such as WordNet are widely used. The integration of a manually engineered knowledge source into the process introduces more

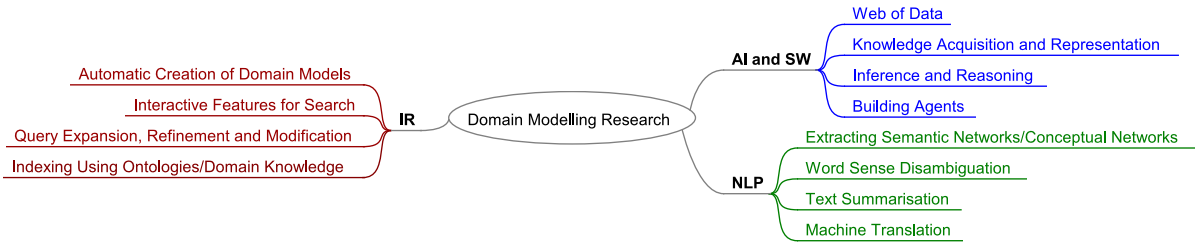


Figure 2.4.: Domain modelling in various disciplines

control over the relationships extracted, but may not be able to cover a sufficient number of domain-specific concepts, which can affect the adaptability of the framework to very specialised domains.

Data-driven approaches have always been widely used in IR. NLP is a prime example of a research field that has seen a shift from mainly symbolic ideas to a strong preference for the statistical approaches which nowadays dominate research. AI and SW technology, on the other hand, is an example where the knowledge-driven approach is more prominent.

Figure 2.4 lists a few examples for each of the three chosen research disciplines. We will now look at these disciplines in more detail.

2.7.1. Artificial Intelligence and the Semantic Web

AI researchers have always been interested in representing knowledge in such a way that it can be utilised by automatic reasoning systems (Sowa, 2008). We can see the idea of the SW as a natural extension to this long tradition.

The main objective of the SW lies in extending the Web to include content currently outside the immediate scope of linked pages, to enable agents to use this content in a variety of applications across different platforms (Berners-Lee et al., 2001). As such, creating common formats and links between databases and their content is at the core of their many tasks. Consequently, domain knowledge representation together with the extraction of fine grained

metadata to describe content form one of the many important areas of research within the SW community. In particular, the ability to extract formal terminology and identify relationships between the terms (a.k.a. *ontology*) from unstructured text is considered to be of critical importance (Buitelaar and Cimiano, 2008; Buitelaar et al., 2005). The PASCAL Ontology Learning Challenge[8] was an example initiative aiming to address this issue, e.g. (Maedche and Staab, 2001, 2004; Navigli and Velardi, 2004).

At the heart of the semantic web is the desire to enable different applications to *understand* and use the same data. This drives domain concepts and relationships between concepts be defined as explicitly as possible. The concepts are often required to express the same level of detail that would be found in a relational database comprising abstractions of inclusion, aggregation and association. This encourages domain models developed within this community to have a strong foundation in knowledge-driven approaches.

The reliance of applications on well-designed data structure leads the research in this community to be largely dominated by semi-automatic and manual approaches, e.g. (Flouris et al., 2008; Maedche et al., 2003). Some tools, however, such as the Karlsruhe Ontology (KAON) framework[9] and OntoLearn (Velardi et al., 2001), actively support language processing for automatically extracting and selecting keywords representative of domain concepts from natural language texts.

Understanding and extracting knowledge from data requires a fine-grained representation of the semantic relationships between entities found within the text. The research in AI and the SW tends to reflect this by focusing heavily on knowledge-driven approaches to domain modelling. Data-driven approaches do, however, also find their way into this area, primarily those that extract relations using NLP methods (Wilks and Brewster, 2006).

[8] http://olc.ijs.si/
[9] http://kaon.semanticweb.org/

2.7.2. Natural Language Processing

Researchers in NLP have shared a long standing interest in constructing domain models or semantic networks to characterise textual structure, to find terms related to each other, e.g. (Ceccato, 1961; Doyle, 1961; Hearst, 1992; Pantel and Lin, 2002; Phillips, 1985; Widdows and Dorow, 2002). A thorough overview of NLP approaches to the construction of conceptual networks can be found in (Widdows, 2004).

A typical data-driven example that illustrates the difference to the AI and SW approaches is introduced by Widdows and Dorow (2002). The algorithm can be used for *"assembling semantic knowledge for any domain or application"*, is based on grammatical relationships such as co-occurrence of nouns or noun phrases, and needs only a corpus tagged for part-of-speech. The underlying motivation is the extraction of term relationships that do not need to strictly follow fully specified semantic relations but which can, for example, be used for query modification in a search context. In other words, the underlying idea is *"to observe word meanings with no prior agenda: to hear the corpus speak with its own voice"* (Widdows et al., 2002).

One example NLP area that profits from the extraction of conceptual graphs from textual documents is word sense disambiguation. It is often an objective in itself in natural language processing, but at the same time it is an essential component in a variety of applications (for example, in question answering). Large-scale conceptual networks have been evaluated and reviewed in the literature as part of the word sense disambiguation task, e.g. (Cuadros and Rigau, 2006) and (Navigli, 2009). NLP techniques have also been used to extract semantic networks (Mintz et al., 2009; Snow et al., 2006), for example, for text summarisation (Lin and Hovy, 2000) and adapting general lexicons to specific domains (Toumouth et al., 2006; Widdows and Dorow, 2002), and so on.

In summary, in the NLP research community, extracting domain knowledge from text is a very active area relevant to a variety of core NLP problems. The main paradigm for current research appears to be the data-driven approach.

2.7.3. Information Retrieval

The IR and Information Seeking (IS) communities aim to build systems that assist users in searching, browsing, and navigating through information spaces. In Section 2.2, we have discussed the added value to the end users of interactive features in IR systems such as query recommendation and faceted search which requires meta-data and additional knowledge about the document collection.

As explicitly engineered ontologies, semantic networks, and document annotation, appropriate for selected domains, are often unavailable and expensive to create, automatically created domain models are increasingly attracting interest within the IR community. Some efforts have already been made to use these query recommendation as discussed in Section 2.4, as well as filtering information, e.g. (Nanas et al., 2009).

The relationships between domain concepts are recognised as being important within IR. For example, networks of hyponym/hypernym relations, and other forms of relatedness have been used to expand, refine, and modify queries and to score document relevance to the given query, e.g. (Grefenstette, 1992; Gürkök et al., 2008; Hovy et al., 2009; Nanas et al., 2009). However, for IR, the emphasis lies in capturing a set of terms that are *closely related* to each other as co-occurring within the same context (whether topical or semantic). As such, the research tends to focus more on clustering concepts for word discrimination rather than on distinguishing between each relationship explicitly to achieve word disambiguation as discussed in (Schütze, 1998).

Like in NLP, the data-driven approach appears to be dominant in IR. However, it is not surprising that some researchers have also turned to formal ontologies that capture domain knowledge (Hsu et al., 2006; Navigli and Velardi, 2003) for query expansion, and have exploited semantic relations from selected texts or queries, e.g. (Hollink et al., 2007; van der Plas and Tiedemann, 2008) for question answering and/or query modification. Vallet et al. (2005), for example, exploited knowledge bases by creating an ontology-based scheme for the semi-automatic annotation of documents and the creation of an IR system using an annotation-weighting and ranking algorithm.

2.8. Concluding Remarks

To conclude this chapter, we found that there is a lot of work done in IR to understand user interactions with a retrieval system and consume them as implicit feedback to either learn a better ranking function or assist future users with the same information needs by recommending queries. However most of these methods target the Web with little work done on enterprise or intranet search which is critical and different from the Web. Moreover the majority of these studies are static and do not incorporate the changes over time that might occur.

On domain modelling, we observe that different research communities have shown a continuing interest in domain modelling, but it appears that there has been little overlap between different disciplines. From the discussion in Section 2.7, it is clear that the approaches adopted by different communities can be complementary to each other. Our contribution fits in the IR community where we develop methods that exploit implicit feedback from users to continuously adapt a domain model that embeds an up-to-date knowledge about a document collection in a domain collectively acquired from the users. Such models can be then used for recommendations, navigation or contextualisation.

Part I.

Building Adaptive Models

Chapter 3.

Domain Modelling with Query Flows

The implicit feedback left by users while they interact with an IR system is rich as it consists of a variety of signals. In this thesis, we propose a number of adaptive domain modelling approaches which use different types of those signals to build a domain model. We split them into two chapters according to the types of implicit signals they employ.

In this chapter, we present modelling techniques which *only* use the sequence (flow) of *queries* identified in individual user sessions. In the next chapter we will discuss how clickthrough data can be further incorporated for adaptive domain modelling.

More specifically, in this chapter we introduce two adaptive domain models derived entirely from query flows found in search logs. They are based on two biologically inspired algorithms.

1. Ant Colony Optimisation (ACO): a swarm intelligence based algorithm which has emerged as a commonly used paradigm to solve practical problems in many disciplines.

2. Nootropia: inspired from the immune system in nature and has been successfully applied in Information Filtering problems such as personalised news recommendation (Nanas et al., 2009).

The rationale behind using biologically inspired algorithms is that they are intuitive and they mimic truly adaptive systems in nature. In particular, both the ACO and Nootropia are easy to implement as they are computationally inexpensive when compared to other biologically inspired algorithms such as neural networks or genetic algorithms. Other adaptive techniques can be investigated to build adaptive domain models such as reinforcement learning (Kaelbling et al., 1996), but we leave that for future work.

Both models rely mainly on using *query reformulations* as implicit feedback from users when they interact with search engines.

First we set the context by formalising the problem and defining the terminology used throughout the thesis. Then we propose the ACO and Nootropia approaches to build domain models from query flows in search logs. We finish the chapter with concluding remarks.

3.1. Terminology and Problem Formulation

In this section, we define the concepts used throughout the thesis, and illustrate the procedures and the assumptions we made to conduct our research and prepare the data. Since query recommendation is a major application of our adaptive domain models that is studied in the thesis, we also formalise the problem of query recommendation.

3.1.1. Basic Concepts

An information retrieval system can be characterised by the following components (Baeza-Yates and Ribeiro-Neto, 2011):

- The *document collection* that is being searched by the user. The finite set of logical representations of the documents indexed by the system is denoted by $\mathbb{D}$.

- The user information needs which are called *queries*. The finite set of logical representations of the user information needs is referred to as $\mathbb{Q}$.
- A mechanism of identifying relevant documents that satisfy the user information needs. This is realised by a ranking function $\mathbb{F}(d_i, q_j)$ defined on $\mathbb{D} \times \mathbb{Q}$ that associates a real number with a query representation $q_j \in \mathbb{Q}$ and a document representation $d_i \in \mathbb{D}$. This would order the documents in the document collection according to their relevance to the user information needs.

Definition 1. ***(Query Recommendation)**: A query recommendation system is a function $\mathbb{R}(q_i, q_j)$ defined on $\mathbb{Q}^2$ that associates a real number with the user's current query $q_i \in \mathbb{Q}$ and another query $q_j \in \mathbb{Q}$.*

This would order queries according to how good they are as alternative queries for the user information needs. By a *good* query recommendation, we mean a query that better represents the user's information needs and yields more relevant documents that satisfy the information needs of the user. We acknowledge that this definition is an approximation as a good query recommendation may depend on the user's task and the context of the user. Note that this is an abstract definition of a query recommender system that can be instantiated on any instance of a retrieval system. Throughout this thesis we develop query recommendation methods for a particular type of retrieval systems, namely text retrieval systems.

3.1.2. Search Logs

Search logs record all interactions performed by the users with a search engine such as the queries submitted and the documents displayed to the user as results and those inspected by the user (clicked), etc. They are usually maintained by search engine providers for a wide range of applications as discussed previously.

A typical *search log* $\mathscr{L}_\tau = \{L_i\}$ consists of a set of records $L_i = \langle q_i, u_i, t_i, V_i, C_i \rangle$ observed in a certain period of time τ bounded by two time stamps t_s, t_e; $t_s < t_e$.

In each record, we store the submitted query q_i, the user identifier u_i, the time stamp t_i of the query, an ordered list of displayed documents to the user V_i, and a set of click events $C_i = \{c_{ij}\}$ where each click event is represented as a tuple $c_{ij} = \langle d_j, t_j \rangle$ holding the document clicked d_j and the time stamp of the event $t_j \in [t_s, t_e]$.

Moreover we refer to the set of users who interact the search engine as $\mathscr{U}$ and the set of documents indexed by the search engine as $\mathscr{D}$. Therefore for each record L_i in the log the user $u_i \in \mathscr{U}$, the displayed documents $V_i \subseteq \mathscr{D}$ and $\forall c_{ij} \in C_i : d_j \in V_i$.

A user search *session* S consists of an ordered sequence of records L_i submitted by the same user within an arbitrary time frame θ.

$S = \langle L_{x_1}, ..., L_{x_k} \rangle$

where $u_{x_1} = ... = u_{x_k} = u \in \mathscr{U}$, $t_{x_1} < ... < t_{x_k}$ and $\forall i \in \{1, 2, ...k-1\}; t_{x_{i+1}} - t_{x_i} \leq \theta$

In the literature of Web search log analysis a common timeout threshold used is $\theta = 30 minutes$ (Jansen et al., 2007). Our definition of a user session based on a certain period of inactivity is founded on previous work for detecting user session boundaries (Gayo-Avello, 2009). It should be noted that automatically identifying the boundaries of user sessions is a difficult task (Göker and He, 2000; Jansen et al., 2007). One of the reasons is that a session can easily consist of a number of *search goals* and *search missions* (Jones and Klinkner, 2008).

For a session $S = \langle L_{x_1}, ..., L_{x_k} \rangle$ containing more than one query ($k-1$ pairs), we consider each pair of consecutive queries $\langle q_{x_i}, q_{x_{i+1}} \rangle$ as a query reformulation. We denote the finite set of all individual user sessions identified in the search log $\mathscr{L}$ with $\mathscr{S}(\mathscr{L}) = \{S_1, S_2, .., Sm\}$.

Definition 2. ***Query Reformulation****: A query reformulation in a search log is a pair of consecutive queries* $\langle q_i, q_{i+1} \rangle$ *submitted in the same user session. We call the first query* q_i *'the reformulated query' and the following query* q_{i+1} *is called 'the reformulation'.*

It is also important to mention the search log statistics that will be used throughout the thesis:

- The frequency of a query $q \in Q$ in a search log which is denoted by $f(q)$.
- The frequency of a query reformulation (q_i, q_j); $q_i, q_j \in Q$ which is denoted by $ff(q_i, q_j)$.

3.1.3. Query Recommendation with Search Logs

Let Q_τ represent all the unique queries in the search log $\mathscr{L}_\tau$. i.e. $\forall L_i \in \mathscr{L}_\tau : q_i \in Q_\tau$. A query recommendation system that relies entirely on the search log can be seen as an extension to Definition 1.

Definition 3. *(**Log-based Query Recommendation**): A log-based query recommendation system is a function $R_\tau(q_i, q_j)$ defined on Q_τ^2 that associates a real number with a query $q_i \in Q_\tau$ and another query $q_j \in Q_\tau$.*

This would order unique queries in the log according to their suitability as alternative queries when a future user of the same search engine submits in the future an identical query to a previously observed query in the log $q_i \in Q_\tau$.

The function R_τ can be developed using a search log $\mathscr{L}_\tau$ recorded within a certain period of time τ bounded by (t_s, t_e); $t_s < t_e$ in order to recommend queries to a user submitting a query at a certain point of time t_k;$t_k > t_e$. It is also possible to update the ranking function with more logs over time which leads to another definition.

Definition 4. *(**Evolving Log-based Query Recommendation**): An evolving log-based recommendation system is developed by updating the ranking function in Definition 3 over time.*

This can be done on a regular basis, e.g. on an arbitrary interval (daily, weekly). Let $\mathscr{L}_{\tau_i}; i = 1, 2, \ldots$ represents a continuously updated search log where $\tau_i = [t_s, t_i] \wedge (t_1 < t_2 < \ldots)$. For convenience we use the notion $\mathscr{L}_i$ rather than $\mathscr{L}_{\tau i}$ to represent a continuously updated

query log and Q_i to represent the growing distinct set of queries found in all batches $\mathscr{L}_j; j = 1,..,i-1,i$.

An evolving log-based query recommendation system is achieved by updating the ranking function using the continuously updated log. The function R_i defined on Q_i^2 represents the evolving query recommendation system. When updating the function, new records are added to the search log and therefore the set of queries Q on which the function operates grow over time.

3.1.4. Domain Models

In all previous definitions of *query recommendation*, developing the function R requires some understanding of the domain of the document collection. This understanding is necessary to be able to recommend alternative query representations to the user which could be more *useful* to find the documents that satisfy the user information needs. We call this a *domain model*. In Section 2.7 we reviewed domain modelling and concluded that it is a broad concept of structuring knowledge embedded in information materials within a specific domain of interest. Note that the term *model* is overloaded as it is used in different disciplines and can a refer to a number of different things. Modelling is a common process in a number of scientific fields such as Mathematics (Aris, 1995), Physics or Computer Science (Embley and Thalheim, 2011). Consequently a domain model can represent various things. For example in software engineering, it may refer to a conceptual high-level understanding of the problem and identifies the relationship between entities and the constraints on the integrity of the system (Embley and Thalheim, 2011). Here we define what a domain model is in the context of the review provided in Section 2.7.

Definition 5. *(**Domain Model**): A domain model $\mathscr{M}$ is an abstract understanding of the document collection in a domain.*

Our focus is on building domain models entirely from the search logs. These models will describe the document collection from the user population's perspective. Note that search logs can also be used to update existing domains.

Definition 6. *(**A Log-based Domain Model**): A log-based domain model $\mathscr{M}_{\mathscr{L}}$ is a model which has been constructed entirely from a query log $\mathscr{L}$.*

Using the same analogy with the previous definitions of log-based query recommendation systems (definitions 3 and 4) we can also define an evolving domain model.

Definition 7. *(**An Evolving Log-based Domain Model**): An evolving domain model $\mathscr{M}_i$ is a model which is updated throughout the time from a continuously updated query log $\mathscr{L}_i$.*

In both previous definitions (6 and 7), the models are independent from the document collection in the domain which can be continuously changing. However we also assume the user queries will reflect this change. We consider this a positive feature of these models as they can adapt to those changes.

In this thesis, we focus on building evolving domain models from search logs. We are particularly interested in making these evolving models *adaptive*. From our perspective, an *adaptive* model should be capable of coping with changes in the document collection or new search trends that may occur as a result of new events in the domain or because of seasonal events, i.e. re-occurring events at certain times in the year.

3.1.5. Text Retrieval Systems

Text Retrieval systems are a special case of IR systems where the document collections are text documents such as Web pages written in natural language and queries are usually expressed by users as natural language words. Web Search engines, we are familiar with, and search engines of local Web sites are examples of these systems. In all our work, we will focus on building domain models for these systems.

In these systems, the document collection $\mathbb{D}$ represents text documents, e.g. HTML Web pages, and queries $\mathbb{Q}$ are expressed by users in natural language as short strings usually containing the keywords that best represent their information needs. Several retrieval models have been studied extensively in the literature to develop the ranking function $\mathbb{F}(d_i, q_j)$ in text retrieval systems such as vector space models (Jones, 1972), language models (Croft and Lewis, 1987), or probabilistic models (Robertson, 1977).

These models rely on an *index* mapping *terms*, defined in a vocabulary, to documents in which they appear in. The vocabulary could be the set of all different words found in the text.

Usually some pre-processing steps are performed on the document text, to build the index, and on the query when applying the ranking function $\mathbb{F}$. These pre-processing steps may include case folding, stemming, stop words removal and so on.

In this work, we focus on building domain models from the search logs *independently* from the retrieval model of the underlying text retrieval system. We mainly consider domains with a text document collection in English and a user base issuing queries in the English language. Nevertheless, this would not alter the conceptual methodologies introduced here. In the following section, we describe the procedure for preprocessing query strings in a query log and our assumptions for building evolving models and query recommendation systems.

3.1.6. Pre-processing Queries for Modelling and Recommendation

Text retrieval systems vary in how they process queries expressed as strings of characters by their users. For example they may perform stemming or stop-word removal. We do not apply the same preprocessing on the queries in the search log, as that performed by the underlying retrieval system, for building domain models or for query recommendation. The intuition is that alternative queries suggested by the models will also be processed by the underlying retrieval system.

However, we do some pre-processing that is routinely performed on queries in the search log for convenience. Here we describe the general procedure applied throughout this work to pre-process queries in search logs.

Let an underscore _ denote a white space character, ASCII punctuations are denoted by P, human language letters are denoted by L and the empty string is represented by λ. Let Σ represents the alphabet of letters, digits and punctuations $\Sigma = L \cup \{0-9\} \cup P$. c_i is a character in that alphabet $c_i \in \Sigma$. w_i is a word in that alphabet $w_i \in \Sigma^*$ and z_i is any string composed from that alphabet or space character $z_i \in (\Sigma \cup \{\_\} \cup \{\lambda\})^*$ including the empty string.

For a search log $\mathscr{L}$ of a text retrieval system, the set of unique queries Q are assumed to be $Q = \{q_i : q_i = z_i \in (\Sigma \cup \{\_\} \cup \{\lambda\})^*\}$.

We define a function $N : Q \times Q'$ for preprocessing or normalising queries in the log $N(z_i) = z'_i$. Q' denotes the set of all unique queries in the log after performing the normalisation on the queries.

The normalisation function N is applied on each query in the logs and consists of the following steps:

- **Punctuation removal**: The punctuation removal can be described as follows.

$$z_i = z_a \; p \; z_b \Rightarrow z'_i = z_a \; z_b \; ; \; p \in P$$

- **Trimming white spaces**: This includes removing leading and trailing white spaces, and reducing middle white spaces into a single white space.

$$z_i = z_s \; z_a \Rightarrow z'_i = z_a \; ; \; z_s \in \{\_\}^*$$
$$z_i = z_a \; z_s \Rightarrow z'_i = z_a \; ; \; z_s \in \{\_\}^*$$
$$z_i = z_a \; z_s \; z_b \Rightarrow z'_i = z_a \; \_ \; z_b \; ; \; z_s \in \{\_\}^*$$

- **Case folding**: This would convert all Latin ASCII characters in upper case to the corresponding lower case character.

$$z_i = z_a\ u\ z_b \Rightarrow z'_i = z_a\ l\ z_b\ ;\ u \in [A-Z] \wedge l \in [a-z] \wedge Lower(u) = l$$

Where *Lower* is a function defined on $[A-Z] \times [a-z]$ that returns for each latin ASCII character in the upper case the corresponding character in lower case.

In the rest of the thesis, we assume that the queries in the logs are normalised using the procedure described i.e. the function N. When we refer to a query in the log q_i we actually refer to the processed query q'_i. Also when we refer to the set of all unique queries in the log Q we actually mean the set of all unique queries after performing the normalisation Q'. In addition to that, the log-based query recommendation system can be seen as function defined on $Q' \times Q'$.

3.1.7. Example of Search Logs

In Chapter 1 we presented our reference system which is the search of an academic institution to provide the reader with motivating examples in which our adaptive models can be useful (See Section 1.1). Here we illustrate how this system record user interactions to give the reader an example of the data that our methodologies work on.

Here is an extract from the actual query log files to illustrate the structure ('xxx' is a field separator):

```
...
1990705 xxx ff39574dafb1944f9c274bcfcc0de329 xxx 2011-04-04 11:34:51  xxx 0 xxx\
0 xxx 0 xxx su xxx su xxx su
1990707 xxx ff39574dafb1944f9c274bcfcc0de329 xxx 2011-04-04 11:34:59  xxx 0 xxx\
      0 xxx 0 xxx students union xxx students union xxx students union
...
```

The logs record a query identifier, a session identifier, the submission time, various forms of the submitted query as well as additional information (not used for our purposes). Displayed

are two interactions submitted within the same session. The query *"su"* is followed by a new query: *"students union"*.

The sample log entries also demonstrate that we do not identify individual users and we do not associate IP addresses with sessions. The underlying reason for that is to comply with data protection issues and to avoid any potential privacy problems. It also fits in with our overall aim of generating recommendations for an entire community of users which is different from personalised recommendation.

The second field in the log record is the automatically generated session identifier which can be seen as an anonymous user ID. It can also facilitate the session detection as the default server timeout is 30 minutes, i.e. a session expires after 30 minutes of inactivity.

The logs also contain information about whether the query is issued by the user or selected from the interactive element of query suggestions. In fact, these queries accounts for around 12% of the queries issued to the search engine. Throughout the thesis we do not make a distinction between a reformulation explicitly entered by the user and a suggestion clicked by the user. The intuition is that the users explicitly made this judgement where they selected the query from a list of other alternatives offered. However, we may want to distinguish between these reformulations and the ones explicitly typed in by the user, but we leave that for future work.

It should be noted that the search engine has been configured such that traffic from other search engine robots is disallowed. This ensures that web crawlers adhering to the Robots Exclusion Protocol [1] do not access (issue queries to) the search engine. Therefore, we have not filtered the logs from potential robot traffic.

Clickthrough data is also available in the logs and can be extracted for each session. Here is an example where we extract clicks for the previous session ('`xxx`' is a field separator):

[1] http://www.robotstxt.org

```
1990705 xxx su xxx 2011-04-04 11:34:51 xxx 2011-04-04 11:34:59 xxx NULL xxx NULL

1990707 xxx students union xxx 2011-04-04 11:34:59 xxx NULL xxx\
http://www.essex.ac.uk/efi/su.htm xxx 2011-04-04 11:35:08
```

Each query in the session will have at least one record. In each record, the following fields are generated: the query identifier, the query itself, the time stamp of the query, the time stamp of the following query in the session if available, the URL clicked and the time stamp of the click event.

The first query "su" has one record where the URL field is NULL which means that no click event has occurred upon submitting the query. The next query "students union" has one record but no follow-up query as the 4^{th} field is NULL. This record corresponds to a click event on the URL provided in the 5^{th} field and its time stamp in the last one. If this query has more clicks more records will be provided corresponding to each click event.

The URLs displayed to the user are also provided in the logs and can be extracted for each query.

Here is an example of where extracted URLs displayed to the user for the query 'students union' ('xxx' is a field separator):

```
1 xxx http://www.essex.ac.uk/phonebook/pages/Students_Union.aspx
2 xxx http://www.essex.ac.uk/efi/su.htm
3 xxx http://www.essexfirst.co.uk/essential/studentsunion.asp
4 xxx http://www.essex.ac.uk/internationalacademy/prospective/campuslife/
survival_guide/livingsocial/su.aspx
5 xxx http://www.essex.ac.uk/ug/student_life/student_union.aspx
6 xxx http://www.essex.ac.uk/pg/student_life/student_union.aspx
7 xxx http://www.essex.ac.uk/ug/student%5Flife/student_union.aspx
8 xxx http://www.essex.ac.uk/pg/student%5Flife/student_union.aspx
9 xxx http://www.essex.ac.uk/vr/tour.aspx?tour=4
10 xxx http://www.essex.ac.uk/students/guide/southend/advice_centre.aspx
```

```
11 xxx http://www.essex.ac.uk/committees/agendas_and_minutes_docs.aspx?committee=USUC
12 xxx http://www.essex.ac.uk/immigration/advice.aspx
13 xxx http://www.essex.ac.uk/records_management/policies/su_policy.aspx
14 xxx http://www.essex.ac.uk/records%5Fmanagement/policies/su_policy.aspx
15 xxx http://www.essex.ac.uk/first/su/index.shtm
16 xxx http://www.essex.ac.uk/vr/tour.aspx?tour=38
17 xxx http://www.essex.ac.uk/vr/category.aspx?cat=7
18 xxx http://www.essex.ac.uk/ldev/induction/induction_talks/colchester/Marianne%20Provan.docx
19 xxx http://www.essex.ac.uk/records_management/policies/gen_SU.aspx
20 xxx http://www.essex.ac.uk/records_management/policies/non_academic_disciplinary_activity.aspx
```

Each record corresponds to a URL displayed to the user along with its position in the displayed list. In the example, the list contains 20 URLs which indicates the user has navigated to the second page of 10 result list.

3.2. Ant Colony Optimisation

Here we will introduce our first approach to build adaptive domain models. Since our purpose is to construct models that are dynamic and evolve over time to adapt to changes in the domain, it seems reasonable to borrow methods from adaptive systems found in nature. In this section, we devise a method for building adaptive domain models with the ant colony optimisation algorithm using query flows found in search logs.

3.2.1. Motivation

Ant Colony Optimisation (ACO) has been studied extensively as a form of swarm intelligence technique to solve problems in several domains such as scheduling (Socha et al., 2003), classification (Martens et al., 2007) and routing problems in telecommunication (Di Caro and Dorigo, 1998).

Inspired by nature, ACO models ants' behaviour in their colonies. Nedjah and de Macedo Mourelle (2006) explains this behaviour as follows. Ants wander randomly, and upon finding food return to their colony while laying down *pheromone trails*. Trails are then followed by other ants and *reinforced* if they find food eventually. However, pheromone trails also *evaporate* over time and hence paths to diminishing food resources become less popular and less visited by ants.

Dignum et al. (2010) presented an implementation of ACO to derive seasonally relevant query recommendations in a university domain. We consider their work as an inspiration for us to further investigate how ACO can be exploited to build adaptive domain models from query flows.

Our approach is to use the ACO analogy to populate and then evolve a directional graph, similar to the one depicted in Figure 1.2, over time. In this analogy the edges in the graph are weighted with the pheromone levels that ants, in this case users, leave when traversing the graph. The resulting model is an *evolving log-based model* as described in Definition 7.

The main intuition of using ACO is its capability of dealing with new emerging situations automatically which in our case maps to new search trends due to temporal or seasonal events and also its power to gradually forget less popular out-dated search routes.

For example, in our reference system (the search engine of the University of Essex), the query 'timetable' is an ambiguous query where the searcher might be looking for the timetable of classes in her department or the timetable of exams. We would expect the latter to be more popular in the academic summer term and the former to be more popular in the academic autumn term when the academic year starts. Another example is the emergence of a new academic conference taking place in the university where users are suddenly interested in searching for the conference but once the conference is over the search trend will suddenly change. The domain model built with ACO should be able to cope well with these situations.

3.2.2. Constructing the Model

The user traverses a portion of the graph by reformulating her queries (analogous to the ant's journey), and the weights of the edges on this route are reinforced (increasing the level of pheromone). Over time all weights (pheromone levels) are reduced by normalising the weights and introducing some evaporation factor to reflect unpopularity of the edge if it has not been used by ants. In other words, we reduce the weight of non-traversed edges over time, to penalise incorrect or less relevant queries. In addition we expect outdated queries to be effectively removed from the model, i.e. the pheromone level will become so low that the query will never be recommended to the user.

Overall this process optimises weights on an evolving directional graph. The growing graph corresponding to a continuously updated search log ($\mathscr{L}_i$; $i = 1, 2..$) is denoted by $G_i = (V_i, E_i, f)$ where

- $V_i \subseteq Q_i$ is the set of nodes which represent a subset of the evolving distinct set of queries in the continuously updated log.
- $E_i \subseteq V_i \times V_i$ is the set of directed edges.
- $f : E_i \rightarrow (0..1]$ is a weighting function that assigns to each edge $(q_a, q_b) \in E_i$ a weight $f(q_a, q_b)$ corresponding to the amount of pheromone deposited on the edge at t_i. t_i represent a point of time corresponding to the end of the batch of search log $\mathscr{L}_i$. We denote this by $w_{ab}(t_i)$.

The pheromone level is updated each time using a new batch of log data $\mathscr{L}_i$. For the edge $q_a \rightarrow q_b$ the pheromone level $w_{ab}(t_i)$ is updated using Equation 3.1

$$w_{ab}(t_i) = N \cdot ((1-\rho) \cdot w_{ab}(t_{i-1}) + \Delta w_{ab}(t_i)) \tag{3.1}$$

where:

- N is a normalisation factor, as all pheromone trail levels originating from a single node are normalised to sum to 1
- ρ is an evaporation co-efficient factor
- $\Delta w_{ab}(t_i)$ the amount of pheromone deposited on the edge $q_a \rightarrow q_b$ when considering the batch of logs $\mathscr{L}_i$. The amount of pheromone deposited should correspond to ant moves on the graph. In our case, this can be the frequency of query reformulations corresponding to the edge. Also the cost of ant moves can be taken into account when calculating the amount of pheromone deposited. Generally it can be calculated using Equation 3.2 (Colorni et al., 1991).

$$\Delta w_{ab}(t_i) = \sum_k \frac{A}{C_k} \text{; For all ants' moves using the edge } q_a \rightarrow q_b \tag{3.2}$$

  where:

  - A is an arbitrary constant, which is chosen to be the average weight.
  - C_k is the cost of (ant k)'s journey when using the edge $q_a \rightarrow q_b$.

Calculating the deposited pheromone in Equation 3.2 is subject to the interpretation of what an ant movement is and how the cost is estimated. We call this the *the pheromone calculation scheme*. We will introduce three different pheromone calculation schemes for Equation 3.2.

To describe the process of evolving the graph with ACO in more detail, we first start with an empty graph $G_0 = (\phi, \phi, f)$.

Then for each new batch of a continuously updated log $\mathscr{L}_i$ we identify all the individual user sessions in that batch $\mathscr{S}(\mathscr{L}_i) = \{S_1, S_2, .., Sm\}$. For each session that contains at least one query reformulation we update the pheromone level on the edges of the graph and add nodes or edges to the graph if necessary using either of the following three pheromone calculation schemes. At the end of processing the batch $\mathscr{L}_i$ we end up with new graph G_i. Note that an

edge is added when any amount of pheromone is deposited between the nodes of the edge, i.e. the amount of pheromone is greater than 0. In a practical application, the weights can be updated offline on a periodic basis, e.g hourly, daily or weekly, or even when a certain number of user sessions have completed. In addition, it is possible to run the algorithm from any point in the user log to any other, this allows us to compare how the model performs for particular time periods.

We will introduce different pheromone calculation schemes to experiment with different interpretations of query reformulations identified in the search sessions. The three pheromone calculation schemes we propose are:

1. **Subsequent Reformulations**: In this case, we will consider only subsequent reformulations found in user sessions. For a session $S = \langle L_k, L_l, L_m \rangle$ containing a query modification chain q_k, q_l, q_m we consider the edges $q_k \rightarrow q_l$ and $q_l \rightarrow q_m$.

   When calculating Δw_{ab} as in Equation 3.2, it will be set to zero if no edges are identified as above. The cost C will be set to 1, and therefore Δw_{ab} will be equal to the frequency of the edges identified multiplied by the arbitrary constant.

2. **Linking All**: In this case, not only subsequent reformulations are taken into account but for each query in a user session, apart from the last one, all following queries in the session are linked to that query. For a session $S = \langle L_k, L_l, L_m \rangle$ containing a query modification chain q_k, q_l, q_m, we consider the edges $q_k \rightarrow q_l$, $q_k \rightarrow q_m$, and $q_l \rightarrow q_m$ as ant movements.

   When calculating Δw_{ab} as in Equation 3.2, it will be set to zero if no edges are identified as above. For the cost C_k, we chose the value of $dist$ where $dist$ is the length between the queries in the session, e.g. for the edge $q_k \rightarrow q_m$ the cost will be 2.

3. **Link to Last**: In this case, we link queries in a user's session to the last query in the session. The motivation of doing this is that the last query in the session might be an indication that the users have found a useful search result. In other words the last query

Algorithm 1: The ACO-based algorithm to build and evolve the domain model.

Input: domain model as a graph G, batch association A_b, number of batches BATCH_NUMS
Output: G

```
1  for b ← 1 to BATCH_NUMS do
2      A_b ← FindAllAssociations(b)
       /* update weights of traversed edges                          */
3      foreach (q, q') ∈ A_b do
           /* Query q' is associated to q in a session in batch b.   */
4          n ← FindNode(G, q)
5          if n = NULL then n ← AddNode(G, q)
6          n' ← FindNode(G, q')
7          if n' = NULL then n' ← AddNode(G, q')
8          e ← FindEdge(G, n, n')
9          τ ← CalculateDepositedPhermone(q, q')
10         if e = NULL then
11             e ← AddEdge(G, n, n')
12             SetWeight(G, e, τ)
13         else
14             SetWeight(G, e, τ + GetWeight(G, e))
15         end
16     end
17     NormaliseAllWeights(G)
18 end
```

may have resulted for example in a landing page. For a session $S = \langle L_k, L_l, L_m \rangle$ containing a query modification chain q_k, q_l, q_m, we consider the edges $q_k \to q_m$, $q_k \to q_m$.

As in the previous scheme we chose the value of *dist* for the cost of the movement.

The ACO procedure is illustrated in Algorithm1. Note that a nominal update value of 1 for the constant A in Equation 3.2 is used for our first log batch $\mathscr{L}_1$, however, any positive real number could have been chosen without affecting the outcome of normalisation.

From now on, we will use 'ACO' to refer to the domain model built by this algorithm as described earlier.

3.2.3. Query Recommendation with ACO

The ACO model can be applied in different IR tasks such as query recommendation, navigation or query expansion. Here we describe how the it can be used to recommend alternative queries for a given query.

The process of recommending queries with the ACO model works as follows. We first find the given query in the model (the graph), then list the connected nodes ranked by the edge weights connecting them to the original query. Indirect associations could also be taken into account. In this case sub-trees can be used to link the original node with nodes further down in the sub-trees.

To understand the value of adding these indirect associations, we will experiment with two different approaches:

1. **Depth 1**: Only direct associations are considered. In this case the evolving recommendation function $R_i : Q_i \times Q_i$ would give a score equivalent to the amount of pheromone deposited between the given queries $R_i(q_a, q_b) = w_{ab}(t_i)$

2. **Depth 2**: Indirect associations are also considered. However only nodes which are linked with a maximum distance of 2 edges away are considered. In this case the weights of these indirect links are calculated as the product of the weights of both edges.

   $R_i(q_a, q_b) = \max(w_{ab}(t_i), \mu_{ab}(t_i))$ where:
   $\mu_{ab}(t_i) = \max_{q_c}(w_{ac}(t_i) \cdot w_{cb}(t_i))| \; \forall q_c \in V_i \; s.t. \; (q_a, q_c), (q_c, q_b) \in E_i$

For example, let us assume that the graph depicted in Figure 1.2 is used to recommend queries for the query ‘timetable office’ using either one of the approaches. With the first approach (depth 1) it will recommend a list consisting of one query ‘exam timetable’. With the second approach (depth 2) it will recommend an ordered list consisting of the queries ‘exam timetable’ and ‘exam dates’ weighted as 0.1 and $0.1 * 0.2 = 0.02$ respectively.

3.2.4. Configuring the ACO Model

As we can see the ACO model as we propose it can be configured with either of the following parameters:

- The evaporation factor ρ which decides how quickly the model forgets the links. A small value for ρ means gradual and slow forgetting whereas a high value would highly reinforce recent observations.
- The 3 different pheromone calculation schemes which affect the pheromone level and the edges identification.
- The depth for recommendation.

In Chapter 6, we will perform a thorough evaluation to study these parameters in the context of query recommendation in a University Web Site.

3.3. Nootropia

Here we will present yet another biologically inspired algorithm to build adaptive domain models. Nootropia is an immune-inspired system originally designed for information filtering (Nanas, 2003). In this section, we propose to use the Nootropia concept to build domain models from query flows in search logs. First we will provide a description of the original Nootropia model and then we explain our contribution where we adopt the concept to build adaptive domain models from search logs.

3.3.1. Description of Nootropia

With Nootropia, a representation of the user's interests, called "profile", is constructed and adapted over time using documents that have been judged relevant, or non-relevant, to the

user's interests. In the case of textual information, the user profile is a weighted term network. A term can be any alphanumeric string (e.g. single word), that can be extracted automatically from a document's text. Terms in this network correspond to antibodies and their weights to antibody concentrations. Links between terms represent antibody-to-antigen interactions and their weights measure the affinity between antibodies. For information filtering, the antibody network is used to evaluate the relevance of each incoming document, through a non-linear, spreading activation process. The profile is adapted over time as a self-organising process, which adjusts the network's structure in response to user feedback. The full algorithm of adapting the profile over time is explained in (Nanas et al., 2009). We used the original implementation of the algorithm provided privately by CE.RE.TE.TH[2] to construct our models. We summarise the algorithm as follows.

The self organising process plays the role of B-cell production by the bone marrow, which introduces new antibody types in the immune repertoire. Given a document with positive feedback, the network's dynamics cause a reinforcement of existing profile terms found in the feedback document (antibodies already in the immune repertoire) by an increase in their weight/concentration. The additional weight (concentration) of a reinforced profile term is subtracted equally from those profile terms that are linked to and not from all profile terms indiscreetly. So, the overall weight of the profile remains constant during the network's dynamics, but a competition between linked terms is introduced. How the weight of profile terms is rearranged depends on the network's structure.

3.3.2. Constructing the Model

The Nootropia system builds an adaptive user profile as a network of correlated terms inferred from feedback documents. We propose to adopt Nootropia to build an adaptive domain model

[2] http://www.cereteth.gr/

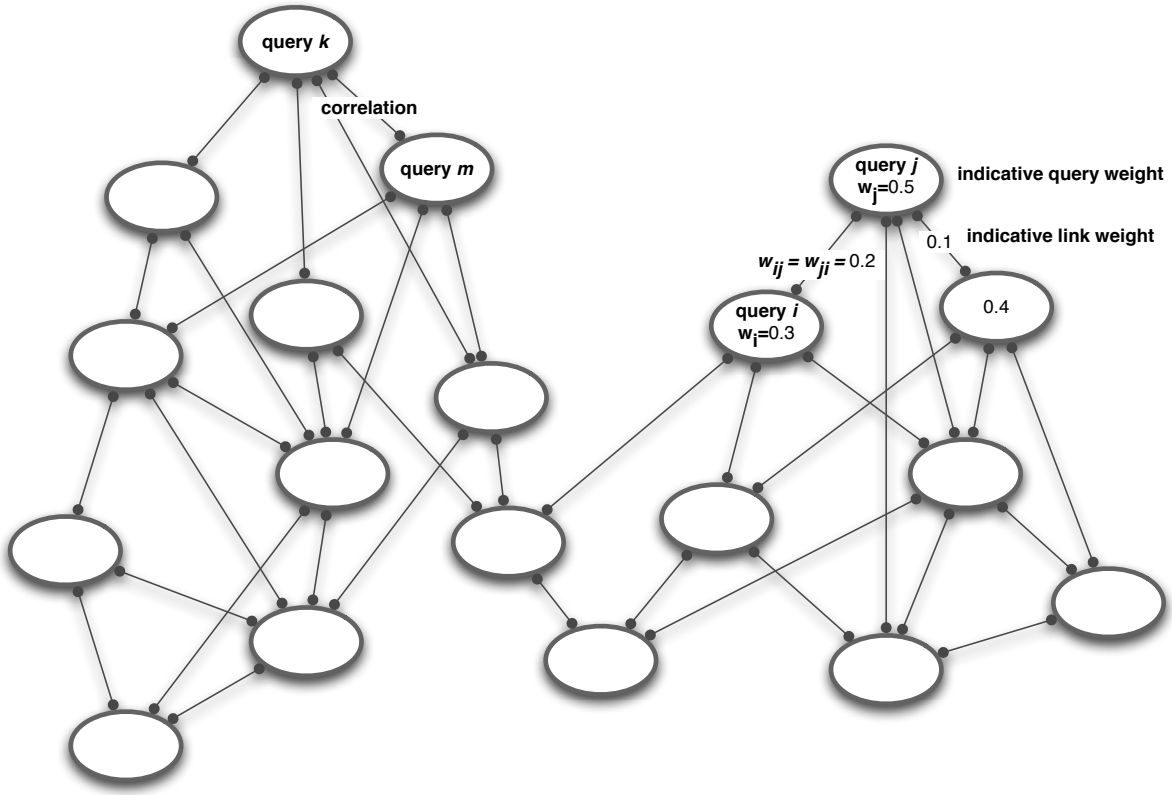


Figure 3.1.: Illustration of Nootropia network.

as a network of correlated queries inferred from the search logs of all users' interactions as illustrated in Figure 3.1.

The main difference to the ACO domain model is that in the case of Nootropia, both nodes and links in the network are weighted. A node's weight indicates the importance of the query within the model and can be statistically calculated.

The process of using Nootropia to build an adaptive domain model as a network of correlated queries from a search log $\mathscr{L}$ can be described as follows. First, we identify all the individual user sessions in the log $\mathscr{S}(\mathscr{L}) = \{S_1, S_2, .., Sm\}$. Each session S_i that contains reformulations is considered as a positive "feedback document" that updates the self-organising network. Each unique query in the session $q_i \in (S) = \{q_1, q_2, .., q_m\}$ receives an increase in weight equal to:

$$w_i = 1/Max(10, m) \tag{3.3}$$

The idea here is to penalise queries that appear in sessions where the user issues many queries. The constant 10 is quite arbitrary, but is common for all queries and hence it does not affect their relative weighting in the domain model, which is the result of the learning process.

To identify and weight the links between query nodes, the *context* of queries is taken into account for measuring co-occurrence data. For this purpose we define two different types of context:

- **Session Windows**: Treating each session as a consecutive series of queries belonging to the same context.
- **Pair Windows**: Breaking sessions into consecutive pairs of queries and treat each pair as the *context* of queries. In other words each query reformulation identified in a session is treated as a context to infer co-occurrence.

To generate the network's links, we treat queries in the same context as correlated and use co-occurrence statistics to calculate the weight of links. We distinguish two types of links:

1. **Non-Symmetric Links**: We take the order of query appearance into account to calculate co-occurrence frequencies. We can then calculate the weight of a link between query q_i and q_j using Equation 3.4. Note that the link's weight is directional ($w_{ij} \neq w_{ji}$).

$$w_{ij} = \frac{(fc(q_i, q_j))^2}{f(q_i) \cdot f(q_j)} \tag{3.4}$$

where:

$fc(q_i, q_j)$ is the number of times q_i is followed by q_j within the same context.

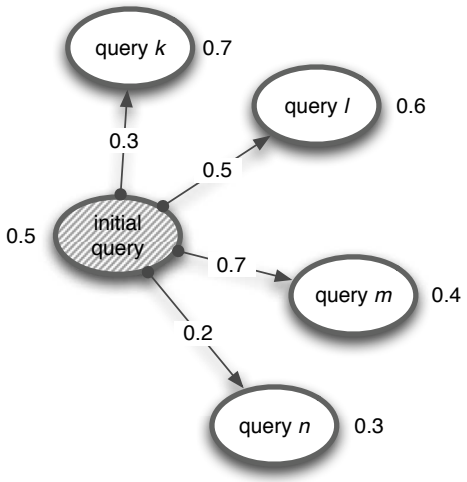


Figure 3.2.: Query recommendation process.

$f(q_i)$ and $f(q_j)$ are respectively the frequencies of queries $q_i, q_j \in Q$ in the search log $\mathscr{L}$.

2. **Symmetric Links**: In this case, we also use Equation 3.4, but we do not take into account the order of the queries in the session ($fc(q_i,q_j) = fc(q_j,q_i)$), and the generated links are not directional ($w_{ij} = w_{ji}$).

The process described here of building the network, i.e. estimating the weights of the nodes and the links, is done on a single batch of search logs. However it can be also extended to a continuously updated search log $\mathscr{L}_i$ as the weights can be recalculated for every new batch using a union of all the previous batches and the current batch $\bigcup_{k=1}^{k=i} \mathscr{L}_k$.

As in ACO, in a practical application, the model can be updated offline on a periodic basis, e.g. hourly, daily or weekly, or even when a certain number of user sessions have completed. In addition, it is possible to run the algorithm from any point in the search log to any other, this allows us to compare how the model performs for particular time periods.

From now on, we will use 'Nootropia' to refer to the domain model built by this algorithm as described earlier.

3.3.3. Query Recommendation with Nootropia

The Nootropia network can be used for query recommendations in the following way. Given an initial query, alternative queries can be recommended if there is a corresponding node in the domain model. Figure 3.2 presents portion of the domain model, which includes the initial query node and its outgoing links to other queries (nodes). In the simplest case, the linked queries can be recommended and ranked according to decreasing link weight, as with the basic ACO approach. In the figure's example, this would result in the following list of query recommendations: query m, query l, query k and query n. However, unlike the ACO approach, with the Nootropia domain model, one can also take into account the additional information that node weights encode. For instance, one could order the recommendations by the product of the link and node weights. In this case, the list of query recommendations would be: query l, query m, query k and query n (different to the previous one).

We propose three different ranking schemes. For a query node q_i of a weight w_i, the model would recommend all the correlated nodes q_j of a weight w_j connected to q_i with a link of a weight w_{ij}, ranked according to one of these ranking schemes:

1. **Nodes weighting**: The correlated nodes are ranked according to their weights w_j.
2. **Links weighting**:The correlated nodes are ranked according to the links w_{ij}.
3. **Nodes+Links weighting**: The correlated nodes are ranked according to $w_j * w_{ij}$.

3.3.4. Configuring Nootropia

The Nootropia model we propose here can be configured with either of the following parameters:

- The definition of context i.e. whether we use session windows or pair windows to decide if two queries have co-occurred.

- The type of links i.e. whether they are directional or non-directional links.
- The three recommendation ranking schemes.

In Chapter 6, we will perform a thorough evaluation to study these parameters in the context of query recommendation in a University Web Site.

3.4. Implementation Issues

Here we discuss some the of the issues that need to be addressed when implementing ACO and Nootropia to provide query recommendations in a live search engine.

The first issue is the scalability. As described earlier updating the graph in ACO or the network in Nootropia is an offline process that can be done on arbitrary equal intervals (e.g. a day or a week). The process involves scanning a new batch of logs, updating the weights on the graph or the network and adding nodes if necessary. This process is not expensive as the complexity is only linear to the number of the nodes and the number of queries in the new batch.

There is also an obvious limitation with the ACO and Nootropia approaches, which is handling new unseen queries. For these queries, both models are incapable of providing any recommendation. In fact, this will be the case for a large proportion of queries when we start with an empty model, i.e when no search history is available (the starting date of operating the search engine). This is also similar to the "cold start" effect in collaborative filtering systems (Schein et al., 2002), where the an item has not received any previous ratings for example because it is new. In such situations, content-based approaches are used to bridge the gap from existing items to new items, by inferring similarities among them. Similarly in our case, the model could be bootstrapped for example by mining associations from the document collection,

e.g. (Kruschwitz, 2003). We have actually investigated a hybrid approach for building adaptive domain models in (Adeyanju et al., 2012a) but it is outside the scope of this thesis.

3.5. Concluding Remarks

In this chapter we introduced two biologically inspired paradigms to turn search logs into useful structures that can be utilised for query recommendation. *Only* query flows submitted within user sessions were used as implicit feedback to create these structures. No clickthrough data or other feedback signals were incorporated in these models

Both models are dynamic and evolve over time to reflect any change in user search behaviour. However both models can be configured in various ways. In the following chapters, we will thoroughly evaluate variations of these models and show how can they adapt over time. Also, we will introduce models where additional information, other than query flows, from search logs such as clickthrough data can be incorporated.

Chapter 4.

Domain Modelling with Clickthrough Data

We have already shown how search logs can be turned into domain models that evolve over time. Only part of the information in the search logs were used to build those models. This part is the query flows identified in individual users' sessions.

However search logs contain richer information which may be also useful to build these structures. The extra information include the post-query browsing behaviour of the user characterised by a number of indicators such as the documents that have been displayed to the user, the documents that have been clicked with their dwelling times and those which have been ignored.

One of the remarkable applications of using the clickthrough data is their ability to automatically evaluate the underlying retrieval model. A number of click-based quality metrics were proposed in (Radlinski et al., 2008) to evaluate the retrieval performance.

In this chapter, we show how clickthrough data can be utilised to enrich the domain models introduced in the previous chapter.

First, we extend a state-of-the-art query recommendation model, the *query flow graph* model, by incorporating the post-query browsing behaviour in the form of click events on displayed results.

Then we show that click-based metrics of the retrieval performance can also be useful to enrich these models by performing an analysis of query reformulations in academic query logs.

4.1. Enriching Query Flow Graphs

The Query Flow Graph (QFG) was introduced in (Boldi et al., 2008) and applied for session identification and query recommendations. Structurally it is similar to the graph we build with the ACO algorithm.

The query flow graph provides an aggregated view of the sequentiality of queries in a large search log aggregated over the user population. However it does not take into account other user interactions than issuing queries.

We want to take advantage of the power of the query flow graph and construct the same structure but embed it with other user actions. This should empower these structures and ultimately adjust query recommendations to user needs.

In this section, we discuss how the graph is built from a search log $\mathscr{L}$ and present our contribution in enriching this model with post-query browsing behaviour in the form of clickthrough data.

4.1.1. Constructing The Model

The query flow graph $G_{qf}(\mathscr{L})$ of a search log $\mathscr{L}$ is a directed graph $G_{qf} = (V, E, w)$ where:

- $V = Q \cup \{s,t\}$ is a set of nodes containing all the distinct queries submitted to the search engine Q and two special nodes s and t representing a *start state* and a *terminate state*;
- $E \subseteq V \times V$ is the set of directed edges;
- $w : E \to (0..1]$ is a weighting function that assigns to every pair of queries $(q,q') \in E$ a weight $w(q,q')$.

The graph can be built from the search logs by creating an edge between two queries q,q' if there is one session in the logs in which q and q' are consecutive. $E = \{(q,q') \,|\, \exists S_j \in \mathscr{S}(\mathscr{L}) s.t. q = q_i \in S_j \wedge q' = q_{i+1} \in S_j\}$

The weighting function of the edges w depends on the application. Boldi et al. (2008) developed a machine learning model that assigns to each edge on the graph a probability that the queries on both ends of the edge are part of the same chain. The chain is defined as a topically coherent sequence of queries of one user. This probability is then used to eliminate less probable edges by specifying some threshold. For the remaining edges the weight $w(q,q')$ is calculated as:

$$w(q,q') = \frac{ff(q,q')}{\sum_{r \in RF(q)} ff(q,r)} \tag{4.1}$$

Where:

- $ff(q,q')$ is the frequency of the reformulation (q,q') in the search log $\mathscr{L}$, i.e. the number of the times the query q is followed by the query q'.
- $RF(q) \subset Q$ is the set of all reformulations of query q in the logs.
  $RF(q) = \{r \in Q \,|\, \exists S_j \in \mathscr{S}(\mathscr{L}) s.t. q = q_i \in S_j \wedge r = q_{i+1} \in S_j\}$

Note that the weights are normalised so that the total weights of the outgoing edges of any node is equal to 1, i.e. $\forall q \in V \,|\, \sum_{r \in N(q)} (q,r) = 1$ where:

$N(q)$: The open neighbourhood of q which is composed of all adjacent nodes to q, i.e the set of nodes $r \in V$ where there is an edge $(q,r) \in E$.

In our work, we adopted the frequency weighting used by (Boldi et al., 2008) without incorporating the learning step as our goal is to show how we can enrich the query flow graph with click data. The learning step can always be added to the enriched version of the graph.

The process described here of building the query flow graph is done on a single batch of search logs. However it can be also extended to a continuously updated search log $\mathscr{L}_i$ as the weights can be recalculated for every new batch using a union of all the previous batches and the current batch $\bigcup_{k=1}^{k=i} \mathscr{L}_k$.

4.1.2. Incorporating Clickthrough Data

In this Section, we explain how we extend the query flow graph model with clickthrough data. The intuition here is to use implicit feedback in the form of clickthrough data left by users when they modify their queries which has been shown to be a powerful feedback, e.g. (Craswell and Szummer, 2007). We consider the number of clicked documents by a user after submitting a query as an indication of the outcome of the reformulation. This indication could be whether it is successful or not, or whether it attracts more interest from the user. This is in line with previous work on evaluating search engines with clickthrough data (Radlinski et al., 2008).

Let $\Phi(q,q') = \langle \varphi_0(q,q'), \varphi_1(q,q'), \varphi_2(q,q'), .. \rangle$ be a sequence of the frequencies of the reformulation (q,q') where:
$\varphi_k(q,q')$ is the number of the times the query q is followed by the query q' in a user session and the user has clicked k (and only k) documents on the result list presented to the user after submitting query q'.

We modify the weighting function in Equation 4.1 to incorporate the click information as follows

$$w(q,q') = \frac{\sum_i C_i.\varphi_i(q,q')}{\sum_{r \in RF(q)} \sum_i C_i.\varphi_i(q,r)} \tag{4.2}$$

where $C = \langle C_0, C_1, C_2, .. \rangle$ is an array of co-efficient factors for each band of click counts. These co-efficient factors will have positive real values $C_i \in \mathbb{R}^+$. Choosing different values for C_i allows us to differentiate between different post-query browsing behaviours, i.e. queries that resulted in different numbers of clicks. For example, queries which result in a single click might be interpreted as more important than the ones which resulted in no clicks or more than one click as the single click may be an indication of quickly finding the document that the user is looking for. Note that navigational queries are usually followed by a single click to navigate to the page (Brenes and Gayo-Avello, 2008). However, here we only consider reformulations of queries. In other words, we try to promote queries which are reformulations of other queries and that have been followed by a single user click.

Later on, We will investigate how different values of the co-efficient C_i affect the the model in its performance quality for query recommendations. Note that the weighting function of the standard graph in Equation 4.1 is the special case where $C_0 = C_1 = C_2 = .. = 1$.

Note we preserve the property of the original QFG model in that for any combination of C_i values, the sum of weights on outgoing edges on a certain nodes will be 1.

$\forall q \in V \mid \sum_{r \in N(q)} (q,r) = 1$

Note that the weighting function specified in Equation 4.2 may result in zero weights. In those cases, edges are eliminated from the graph.

4.1.3. Query Recommendation with QFG

The query flow graph can be used for query recommendation by ranking all the nodes in the graph according to some measure which indicates how reachable they are from the given node (query). We already presented a simple method for recommending queries with a similar graph structure we create with the ACO approach.

Boldi et al. (2008) proposed to use graph random walks for this purpose and reported the most promising results by using a measure which combines relative random walk scores and absolute scores. This measure is

$$\bar{s}_q(q') = \frac{s_q(q')}{\sqrt{r(q')}} \tag{4.3}$$

where:

- $s_q(q')$ is the random walk score relative to q i.e. the one computed with a preference vector for query q.
- $r(q')$ is the absolute random walk score of q' i.e. the one computed with a uniform preference vector.

In our case study described later, we adopted this measure for query recommendation and used the random walk parameters reported by (Boldi et al., 2008). In this case the ranking function used for recommendation is specified in Equation 4.4:

$$R_{QFG}(q,q') = \bar{s}_q(q') \tag{4.4}$$

The recommendation process with random walk is computationally expensive when compared to suggesting neighbouring nodes. It involves converting the graph with n nodes into a raw

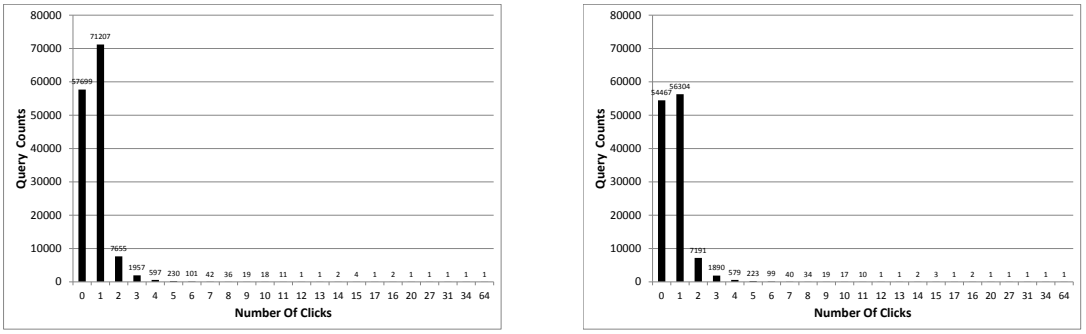
probability matrix of a size $n \times n$ and then performing a series of matrix multiplication until the weights converge. Therefore when the graph grows, the processing time increases exponentially. The time complexity of the recommendation algorithm described is is $O((n+1) \cdot n^2)$.

4.2. Case Study of Enriched Query Flow Graphs

The aim of the case study is to understand the impact of different values of the co-efficient factors of click counts presented in Equation 4.2 in the context of query recommendations.

For this case study, we use a portion of our main dataset, the University of Essex search logs, described earlier in Section 3.1.7. We discuss the variations of QFG models we would like to test with this dataset and illustrate with examples the impact of each variation.

4.2.1. The Dataset



(a) All records in the log. **(b)** Top 5 queries omitted.

Figure 4.1.: Frequency of queries for each band of click counts.

As we have seen, the University of Essex search logs presented in Section 3.1.7 contains all the information usually found in typical search logs, i.e. the queries that have been submitted by a user in a single session, the documents displayed to the user and the ones which were clicked.

A period of 10 weeks of logs between 24 February 2011 and 3 May 2011 inclusive was used for this case study. We denote this portion of search log by $\mathscr{L}_{SX11}$. During this period, the total number of queries submitted to the search engine is $|\mathscr{L}_{SX11}| = 139,588$. Note that these are not unique queries. The number of sessions identified in the log is $|\mathscr{S}_{SX11}| = 89,046$ user sessions and the total number of clicks on the results that were logged $\sum_i |C_i = \{C_{ij}\}| = 97,924$.

Figure 4.1a illustrates a histogram of the frequency of queries corresponding to the resulting number of clicks following each query as recorded in the logs of that search engine. We see that most queries either results in one click or no clicks at all. Removing the top 5 frequent queries in the log which are navigational queries, e.g. 'library' and 'moodle', we show the resulting histogram in Figure 4.1b. The picture is similar but with a lower count of 1-click queries.

4.2.2. Variations of Enriched Query Flow Graphs

We want to study the impact of various combinations of the co-efficient factor of click counts on the nature and quality of the query recommendations generated by the model.

In Figure 4.1a, we can see that only a small fraction of queries end up with more than two clicks. Based on the fact that less than 2% of all queries result in more than 2 clicks, we simplified Equation 4.2 as follows:

$$w(q,q') = \frac{C_0.\varphi_0(q,q') + C_1 \cdot \varphi_1(q,q') + C_k \cdot \varphi_k(q,q')}{\sum_{r \in RF(q)} \sum_i C_i \cdot \varphi_i(q,r)} \tag{4.5}$$

where C_k is the co-efficient factor of all click counts which are larger than 1. i.e. no matter whether a query has resulted in 2 or more clicks on resulting documents we treat all cases the same.

Table 4.1 lists all the combinations we considered and further evaluated in Chapter 6. $QFG_{standard}$ is the standard query flow graph where no click information is incorporated.

QFG_{no_zero} is an enriched query flow graph where reformulations which result in no clicks on the presented document list to the user are not considered. Both QFG_{boost_one} and $QFG_{boost\_one\_more}$ are enriched graphs that boost queries with a single click on the presented list. $QFG_{penalise\_many}$ penalises queries which attract 2 clicks or more. The intuition here is that these queries may have retrieved results that require more effort from the user to successfully find a page that satisfies her information needs. However, this is only an assumption we make and it will be tested in our evaluation. The chosen combinations will test different hypotheses for interpreting the number of clicks. Other combinations may also be considered to test different hypotheses.

	C_0	C_1	C_k
$QFG_{standard}$	1.0	1.0	1.0
QFG_{no_zero}	0.0	1.0	1.0
QFG_{boost_one}	1.0	2.0	1.0
$QFG_{boost\_one\_more}$	1.0	3.0	1.0
$QFG_{penalise\_many}$	1.0	2.0	0.5

Table 4.1.: Experimental graphs.

It is important to mention that we adopted the frequency weighting used by (Boldi et al., 2008) without incorporating the learning step as our goal is to show how we can enrich the query flow graph with click data. Also incorporating the learning step requires the learnt model which is not publicly available. However, the learning step can always be added to the enriched version of the graph.

4.2.3. Examples

After building the graph as explained in the previous section with the search logs $\mathscr{L}_{SX11}$, we randomly selected some queries from the the top 100 queries in the log. We then generated the recommendation list for each of those queries using the random walk scores as described in Section 4.1.3. We show the queries and the 10 recommendations provided by each of the graphs

$QFG_{standard}$, QFG_{boost_one}, $QFG_{boost\_one\_more}$, QFG_{no_zero}, and $QFG_{penalise\_many}$ in Tables 4.2, 4.3, 4.4, 4.5, and 4.6. The examples presented here are only for illustration purposes and do not serve as an evaluation.

In all the graphs, we see that the same query is on the top of the list for each query. It is the most likely query that the random walker will end up at as described in the recommendation process. This is because the parameter used for random walk by (Boldi et al., 2008) that controls the probability of the random walker to stay at its current node is $\alpha = 0.85$ (a common value in the PageRank literature (Bianchini et al., 2005)) and the probability that walker to move to a new node is $1 - \alpha = 0.15$. Note that the letter 'T' refers to the terminate state node.

With a quick glance at all the tables, we can see that most recommendations are domain specific. One can infer from these recommendations that the context here is a university domain. This indicates the power of these methods to adapt to the domain of any given search log.

However, it is difficult to judge whether a recommendation is useful for the query or not, especially as these recommendations are specific to the University of Essex domain. For example, the query 'sub zero' in the University of Essex domain is very different from the the Web in general. It most likely refers to a night club run by the student union at the university and therefore could be judged as a useful query recommendation.

Now we examine the recommendations generated by each graph. In Table 4.2 we see that the original graph $QFG_{standard}$, i.e. the graph that does not incorporate click information, suffers from noise. Across the four lists we observe what can be considered irrelevant alternatives in the University of Essex domain or even misspellings. For example, 'course' $\rightarrow$ 'viology', 'student union' $\rightarrow$ 'introduction', 'law' $\rightarrow$ 'i finish third year' , and 'registry' $\rightarrow$ 'tonu rich'. Those examples were eliminated in all other graphs where we adjusted the weights on the edges depending on the reformulation outcome.

In Tables 4.3, 4.4 both QFG_{boost_one} and $QFG_{boost\_one\_more}$ are very similar but we see that they have eliminated the previous examples of noise and introduced some new (good) alternatives up in the ranking, e.g. 'student union' $\rightarrow$ 'student union officers', 'course' $\rightarrow$ 'coursefinder', 'law' $\rightarrow$ 'undergraduate law llb', and 'registry' $\rightarrow$ 'letter schengen'.

QFG_{no_zero} examples in Table 4.5 look less similar to the previous two graphs. As before it eliminated some noise and introduced new useful alternatives which were not in the previous three, e.g. registry $\rightarrow$ 'viva', 'registry' $\rightarrow$ 'certificate transcripts' and 'student union' $\rightarrow$ 'societies'.

The examples in Table 4.3, 4.4 for $QFG_{penalise\_many}$ are more similar to the QFG_{boost_one} and $QFG_{boost\_one\_more}$ examples.

courses	**student union**	**law**	**registry**
courses	student union	law	registry
undergraduate courses	students union	T	alison booth
T	entertainments	kian lowe	T
degree	T	law degree	esf
online courses	introduction	department of law	regidtry
intruduction forensic science	sub zero	philosophy politics economics	reserach team registry
biochemistry degree	student union book shop	i finish third year	7 day late submission
viology	ents	exams timetablr	hedi
undergraduate programmes	school s mini bus	criminal law revision	tonu rich
courses southen sea	student union mondo	law requirements	student counselor

Table 4.2.: Examples of recommendations by the query flow graph $QFG_{standard}$.

courses	**student union**	**law**	**registry**
courses	student union	law	registry
undergraduate courses	students union	department of law	alison booth
degree	entertainments	law degree	grad office mail id
courses southend university	student union officers	T	letter schengen
option course	sub zero	law requirements	registry emma
criminology department	flirt	foreign student tuition	certificate transcripts
coursefinder	bars	law politics module	post graduate admissions
englis course	introduction	undergraduate law llb	enterprise office
undergraduate course	faculty convenors	result examination	T
online courses	T	kian lowe	planning office

Table 4.3.: Examples of recommendations by the query flow graph QFG_{boost_one} .

courses	student union	law	registry
courses	student union	law	registry
undergraduate courses	students union	department of law	alison booth
courses southend university	student union officers	law degree	grad office mail id
option course	flirt	law requirements	letter schengen
criminology department	bars	foreign student tuition	registry emma
coursefinder	faculty convenors	law politics module	certificate transcripts
englis course	sub zero	undergraduate law llb	post graduate admissions
undergraduate course	entertainments	result examination	enterprise office
biological	introduction	llm law	planning office
degree	T	accommodation 2009 2010	T

Table 4.4.: Examples of recommendations by the query flow graph $QFG_{boost\_one\_more}$.

courses	student union	law	registry
courses	student union	law	registry
undergraduate courses	students union	department of law	alison booth
biological	faculty convenors	law requirements	grad office mail id
journalism courses	student union officers	llm law	letter schengen
courses southend university	entertainments	foreign student tuition	registry emma
criminology department	bars	law politics module	post graduate admissions
option course	flirt	undergraduate law llb	enterprise office
clinical doctorate	introduction	result examination	certificate transcripts
design	transfer	law degree	planning office
undergraduate course	societies	accommodation 2009 2010	viva

Table 4.5.: Examples of recommendations by the query flow graph QFG_{no_zero} .

courses	student union	law	registry
courses	student union	law	registry
undergraduate courses	students union	law degree	alison booth
degree	student union officers	department of law	grad office mail id
courses southend university	sub zero	T	letter schengen
option course	flirt	law requirements	registry emma
criminology department	bars	foreign student tuition	certificate transcripts
coursefinder	entertainments	law politics module	post graduate admissions
englis course	faculty convenors	undergraduate law llb	enterprise office
undergraduate course	T	result examination	T
online courses	introduction	kian lowe	planning office

Table 4.6.: Examples of recommendations by the query flow graph $QFG_{penalise\_many}$.

Overall we see from the examples that incorporating click information affected the recommendation process as the graphs return different rankings and different items in those rankings. We also see from the examples that incorporating the clicks in any of the suggested forms reduced the noise and proposed useful alternatives, i.e. incorporating the post-query browsing

behaviour added value to the graph. The different variations resulted in different rankings and different recommendations but not in all the cases.

However we do not present these examples as a proof of the usefulness of incorporating clicks. Nor do we claim to understand the difference between each variation of the graphs. We are just presenting a case study to give the reader a flavour of what the results look like. We will do a thorough evaluation of these graphs later in Chapter 6.

4.3. Beyond the Number of Clicks

So far, we have shown how clickthrough data can be used to add value to our evolving models in a simplistic approach where only the number of clicks after a reformulation are considered. However click data is much richer as it contains the documents that have been clicked and those which were ignored among other information.

In Chapter 2, we have shown an extensive list of examples where such implicit feedback left by users serves a number of tasks in IR, e.g. Learning to Rank, Automatic Evaluation, etc. One remarkable application is the capability of clickthrough data to automatically assess the retrieval performance of the underlying IR system. With this approach no expensive relevance judgements are required to evaluate a retrieval model in the traditional IR Cranfield evaluation paradigm which dominated IR research for a long time (Vorhees and Harman, 2005). A number of retrieval quality measures estimated from click data has been studied previously in (Radlinski et al., 2008).

In this section, we discuss how these measures can guide the process of building adaptive models to further enhance to the ones we have built so far. First we introduce what we call 'click-based quality measures', then we show how these measures can be utilised for query recommendation by performing an analysis on our main dataset.

4.3.1. Click-based Quality Measures

In our analysis, we experimented with three different click-based measures. They were found to correlate with retrieval performance in that they get worse when the retrieval performance degrades and vice versa (Radlinski et al., 2008).

To define these measures, we take a typical search log $\mathscr{L}_\tau = \{L_i\}$, as defined in Section 3.1.2, that consists of a set of records $L_i = \langle q_i, u_i, t_i, V_i, C_i \rangle$. The following measures are considered:

1. *Mean Clicks* $Mc(q)$ for query $q \in Q$: The mean number of results that are clicked for each query.

$$Mc(q) = \frac{\sum_i |C_i|}{f(q)} \; ; \forall L_i \mid q_i = q \tag{4.6}$$

2. *Max Reciprocal Rank* $MaxRR(q)$ for query $q \in Q$: This is the mean value of $1/r$, where r is the rank of the highest ranked result clicked on after each time the query q is issued.

3. *Mean Reciprocal Rank* $MRR(q)$ for query $q \in Q$: The mean value of $\sum_i 1/r_i$ of all clicks on the result list after each time the query q is issued.

The click-based quality measures are good candidates to judge whether the retrieval system is capable of satisfying a user query. For each query it gives an indication of how 'successful' the query is over an entire given period of search logs. In this context, 'successful' means taking a user to (potentially) matching documents. Here we propose that these measures can be taken forward and guide the process of building better query recommendation systems. For that purpose we conduct an analysis on our main dataset where we observed the query reformulations in the search log. First, we study the overall effectiveness of the retrieval performance of the reformulations estimated by the click-based quality measures. This would validate our previous

proposals of relying on users to build query recommendations systems. Then we employ the click-based quality measures to study an important factor in query recommendation systems, which is the popularity of the query. Using the click-based measures we study the relationship between the popularity of a reformulated query and the observed increase or decrease in the retrieval performance. Finally we would like to see how these measures can be incorporated in our adaptive models for query recommendation.

4.3.2. Log Data Analysis

We perform an analysis on our dataset $\mathscr{L}_{SX11}$. The full specification of the dataset $\mathscr{L}_{SX11}$ was provided before in Section 4.2.1. We extract the set of all the query reformulations found in this log using the Definition 2. A total number of 38,940 reformulations were extracted identified by an automatically generated session identifier (a session expires after 30 minutes of inactivity). The 30 minute timeout is a common threshold used in the literature of Web search log analysis (Jansen et al., 2007). For each reformulation $\langle q_i, q_{i+1} \rangle; q_i = q \wedge q_{i+1} = r \wedge q, r \in Q$ and to estimate the reformulation effectiveness we calculated the following differences: Δf, ΔMc, ΔMRR and $\Delta MaxRR$ of the measures discussed in the previous section and calculated over the entire search log between the reformulation r and the original query q, e.g. $\Delta f = f(r) - f(q)$.

The first section in Table 4.7 summarises the overall user population behaviour in reformulating their queries in terms of the difference in frequency and the difference of retrieval performance estimated by a number of click-based measures. Looking at the first row, we observe that users reformulate to more frequent queries as we can observe an average increase in the frequency of the query. Also taking any of the click-based measures we can see an overall increase in the performance suggesting that users succeed in choosing more useful reformulations. In the second row we take a subset of the reformulations to eliminate reformulations to navigational queries. For example the query 'moodle' has a frequency of 10,872 in the log and there are a lot of cases where users reformulate to 'moodle' from a misspelled version

(e.g. 'moddle'). Therefore we look at only reformulations where the difference in frequency between the two queries is less than 100. The same observation holds here with an increase on all the measures. Further analysis is illustrated in Figure 4.2 (top), where the percentage of the increase or decrease is shown across the various click-based measures. Overall the positive differences are dominant.

	Count	Δf	ΔMc	ΔMRR	$\Delta MaxRR$
All cases	38,940	+74.080	+0.082	+0.056	+0.065
-100< Δf <100	29,180	+0.478	+0.066	+0.042	+0.049
Δf <0	14,049	-454.741	-0.229	-0.245	-0.160
-100< Δf <0	9,960	-21.319	-0.214	-0.248	-0.136
-50< Δf <0	8,289	-11.652	-0.224	-0.257	-0.142
-10< Δf <0	5,253	-3.581	-0.236	-0.244	-0.141
Δf >0	14,950	+590.096	+0.394	+0.356	+0.293
0< Δf <100	10,052	+22.256	+0.366	+0.344	+0.248
0< Δf <50	8,397	+12.391	+0.374	+0.345	+0.249
0< Δf <10	5,100	+3.846	+0.383	+0.331	+0.242

Table 4.7.: Averages of the differences in frequency and the differences in the performance estimated by 3 click-based measures across all reformulations.

To study the relationship between the popularity of a reformulated query and the observed change in retrieval performance, we split the reformulations into reformulations to less popular queries $\Delta(f) < 0$ and reformulations to more popular queries $\Delta(f) > 0$. In the second section of Table 4.7, we show the results for the former. Overall this strategy seems to fail with an average decrease across all the measures. As before we tested on a different subset $-100 < \Delta(f) < 0$ to eliminate the effect of navigational queries and the results remain negative. All differences in all measures are significant using 2-tailed t-tests ($p < 0.0001$). In the third section of Table 4.7, we show the results for the reformulations to more popular queries, overall this strategy seems to succeed with an average increase across all the measures. All differences in all measures are significant using 2-tailed t-tests ($p < 0.0001$). In Figure 4.2 (medium, bottom), we can see that

the increases of measures are dominant for reformulations to more popular queries $\Delta(f) > 0$ and the decreases are dominant for reformulations to less popular queries $\Delta(f) < 0$.

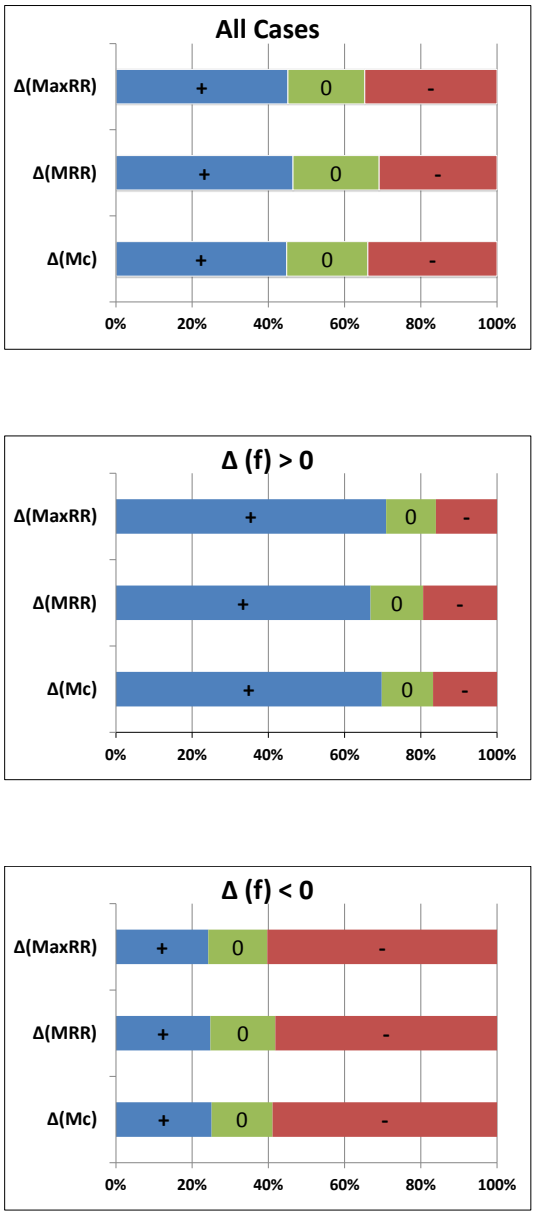


Figure 4.2.: Percentage of reformulations with increase(+), no change(0) or decrease(-) of the various measures in three different subsets.

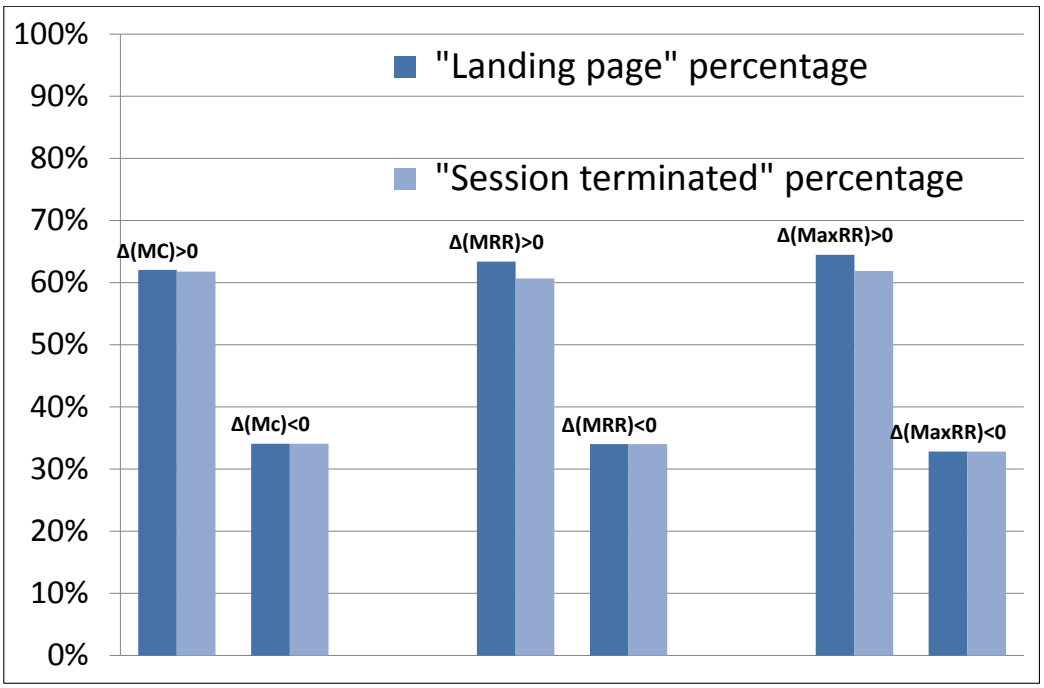


Figure 4.3.: Percentage of reformulations followed by either a single landing page or a termination of the session across different categories (increase/decrease in each click-based measure).

Finally, to see whether click-based quality measures are able to identify useful reformulations, we split the reformulations according to the increase/decrease across the various measures and observe two different criteria for a usefulness of the reformulation:

- The reformulation was followed by a landing page after a single click.
- The session has terminated.

In Figure 4.3, we can see that when the measure gets better (positive increase), it often results in meeting both criteria and vice versa.

4.3.3. Summary of the Reformulation Analysis

Assuming the sample domain is typical, we find that users of a university search engine are able to reformulate their queries to more successful ones and not surprisingly that reformulations to more popular queries are more successful than less frequent ones. We also show the ability of click-based quality measures to identify useful query recommendations. This analysis suggests that our adaptive models which rely on query reformulations are valid since we can observe an overall success over the user population. Moreover, the analysis suggests that an increase in either of our click-based quality measures should translate in an adaptive model as a positive reinforcement.

4.4. Concluding Remarks

In this chapter, we presented another source of implicit feedback found in search logs to build evolving domain models for query recommendation. Post-query browsing behaviour embedded in the form of click data was employed to enrich the adaptive models. We proposed to use a intuitive approach to extend a state-of-the art model by using only the number of clicks upon submitting a reformulation. Also a more elaborate interpretation of click data was considered to guide the process of building adaptive models.

Part II.

Evaluation

Chapter 5.

Automatic Evaluation of Adaptive Query Recommendation

Assessing the effectiveness of retrieval systems by conducting user studies is a common practice (Kelly, 2009). Although these studies provide a comprehensive qualitative analysis of the retrieval effectiveness, they are generally expensive. The problem is even more pronounced when assessing adaptive search systems, for example evolving query recommendation systems like those presented in the previous two chapters. Automatically predicting the performance of evolving domain models in producing useful query recommendations *before* getting the users involved is therefore highly desirable. In this chapter, we propose *AutoEval*. *AutoEval* is an evaluation methodology that assesses the quality of query recommendations generated by a domain model by entirely relying on past user interactions with the system.

5.1. Why Automatic Evaluation?

As we have seen in Chapter 2, generating relevant query recommendations has been a research topic in IR for years and a number of methods have been developed to derive query recommendations such as (Boldi et al., 2008; Kruschwitz, 2005; Sanderson and Croft, 1999). In Chapters 3 & 4, we have also presented a number of methods for building evolving domain models for query recommendation.

However, the evaluation of query recommendations has remained a major challenge partly due to the lack of suitable test collections. We have already identified few approaches for evaluating query recommendation systems in Section 2.4.5. Generally, we can classify these approaches into:

- *system-oriented* approaches where we assess the effectiveness of the system using ground-truth data (e.g. existing knowledge sources, or collected explicit judgements);
- *user-oriented* approaches where we conduct controlled user studies and assess how satisfied the user is with the system.

Although user studies have the power of directly obtaining explicit feedback from the end user and providing qualitative explanations, they tend to be expensive. Moreover, they do not allow an exploration of the multitude of parameters that are often needed to be tuned in machine learning systems which would require a large number of iterations in the evaluation process. Furthermore, they can be affected by a great deal of subjectivity in the users' perceptions. Therefore, it is desirable to be able to perform some offline experiments where these models can be tested and their performance is observed before going live.

On the other hand, system-oriented evaluations of retrieval systems rely on explicit user judgements that also need to be collected on a large scale. However, the automatic evaluation

of search systems that does not rely on expensive user judgements has long been attracting IR researchers, e.g. (Soboroff et al., 2001) and we aim to investigate this avenue in our work.

In the previous chapters, we have proposed adaptive query recommendation systems that do not produce a static set of recommendations but constantly learn and evolve using query logs in a continuous learning process. It is highly desirable to test and explore different approaches for adaptive query suggestions *before* employing a technique in a live setting. This is however not an easy exercise and unlike commonly understood TREC measures (such as Mean Average Precision), there is no commonly agreed automatic evaluation measure for *adaptive* search. In recommender systems, the evaluation of collaborative filtering techniques have traditionally focused on static collections (Adomavicius and Tuzhilin, 2005) where these methods are trained and tested on static datasets which capture user preferences at a certain point of time. Recently, there have been some attempts to evaluate recommender systems over time (Lathia, 2010) where user preferences and item popularities dynamically change over the course of evaluation. We aim to take this concept forward and evaluate our adaptive query suggestions in a dynamic environment.

In this chapter, we introduce a new evaluation approach based on search logs. Search logs contain information about what users have entered and clicked. It is a reflection of a reality and is representative to both its document collection and its search transactions. *AutoEval* is a methodology that performs simulated query recommendation experiments based on past log data to evaluate different models for generating query suggestions.

5.2. The AutoEval Methodolgy

Our automatic evaluation framework assesses the performance of query recommendation systems over time using actual query logs by comparing suggestions derived from a domain

model to query reformulations actually observed in the log files. The intuition is that query logs are a reflection of reality and a better system should be able to perform better in a real scenario.

The evaluation works as follows. Let $\mathscr{L}_i$ represents a continuously updated search log, as defined in Section 3.1.2. The model's evaluation is performed on each batch of the continuously updated log. First we extract all the sessions in the batch $\mathscr{S}(\mathscr{L}_i) = \{S_1, S_2, .., Sm\}$. Then, for each session we identify the query reformulations found if any. For each identified query reformulation (q, q'), the domain model is provided with the reformulated query q and returns a ranked list of recommended queries. We take the rank of the actual *reformulation* q' (i.e., the one in the log data) in this list, as an indication of the domain model's accuracy. The assumption here is that the reformulations in the logs represent the 'ground truth'. We acknowledge that user's reformulations may be poor or incorrect resulting in noisy reformulations. Our assumption is indeed an approximation but it allows us to conduct large-scale experimentation. Moreover, the used session identification method has an impact on the quality of the reformulations identified in the search log. Our definition of a session in Section 3.1.2 does not guarantee that subsequent queries in a user session are actually related. However, we argue that those noisy query reformulations do not affect the evaluation methodology as this noise is common for all evaluated model.

Also we are making an assumption that an accurate domain model should be able to propose the most appropriate alternative query at the top of the list of recommended queries. This is based on the observation that users are much more likely to click on the top results of a ranked list than to select something further down (Joachims et al., 2005), and it seems reasonable to assume that such a preference is valid not just for ranked lists of search results but for lists of query recommendations as well.

For example, let us assume that during the current batch, three query reformulations have been submitted (as depicted in Figure 5.1). So for the total of three query reformulations in the current batch, we can calculate the model's Mean Reciprocal Rank (*MRR*) score as

$(1/r_1 + 1/r_2 + 1/r_3)/3$, where r_1 to r_3 are the ranks of the actual reformulation in the list of queries recommended by the model in each of the three cases.

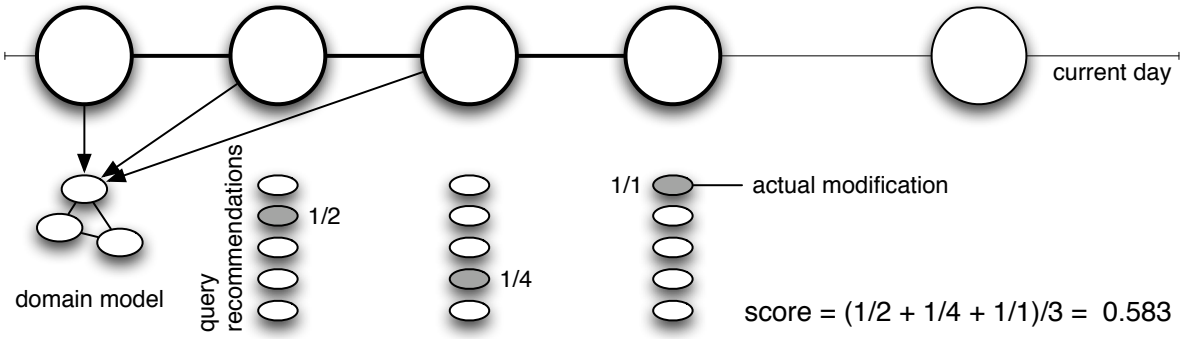


Figure 5.1.: The evaluation process

More generally, given a batch $\mathscr{L}_i$ with a total n query reformulations, the model's Mean Reciprocal Rank score for that batch MRR_i is given by Equation 5.1.

$$MRR_i = (\sum_{j=1}^{n} \frac{1}{r_j})/n \tag{5.1}$$

Note that in the special case where the actual reformulation is not included in the list of recommended queries then $1/r$ is set to zero. The above evaluation process results in a score for each log batch. So overall, the process produces a series of scores for each domain model being evaluated. These scores allow the comparison between different domain models. A model M_1 can therefore be considered superior over a model M_2 if a statistically significant improvement can be measured over the given period.

Note the assumption that the top ranked query suggestion is the most relevant may not hold in situations where query recommendations are presented in a non-list manner. Moreover, it is not suitable when the diversity of query recommendations is taken into account, i.e. recommending diverse queries in favour of relevant but redundant queries. Diversity has been studied in the recommender system literature e.g. (Adomavicius and Kwon, 2012) and also in

query recommendation (Zhu et al., 2010). In these situations, other evaluation criteria were used, for instance Bradley and Smyth (2001) used item dissimilarity between the recommended items for measuring the diversity and use that as one of their quality metrics. In our work, as we are focusing on the adaptivity aspect of recommendations we have not considered diversity and we leave that for future work.

Following this, we also consider other metrics to study different aspects of the evaluated models. One other metric that can be used is the success rate at a certain rank. We define the success rate at rank k ($SR@k$) as the percentage of query reformulations within a certain interval where the model succeeded in providing the reformulated query in its top k suggestion list.

$$SR@k_i = (\sum_{j=1}^{n} success(j))/n \tag{5.2}$$

where

$$success(j) = \begin{cases} 1 & \text{if } 0 < r_j \leq k \\ 0 & \text{otherwise} \end{cases}$$

Such a metric was used in (Harvey et al., 2010) for validating a system that provides personalised tags.

The described process fits a static model perfectly, but in the case of evolving log-based domain models as those we are experimenting with here, the experimental process is similar. We start with an initially empty domain model. Like before, the model is evaluated at the end of each batch of a continuously updated search log, but unlike the static experiments it uses the batch log for updating its structure. In other words, for every new batch of a continuously

updated log $\mathscr{L}_i$, the evaluation score(s) is calculated for a model already trained on all the previous batches, i.e. the search log $\bigcup_{k=1}^{k=i-1} \mathscr{L}_k$. Then the batch is used to update the model.

In the next section, we aim to validate the framework by conducting a user study in parallel with applying AutoEval on a number of evolving log-based domain models.

5.3. Validation of AutoEval

The aim of the experiment is to find out whether the performance predicted by *AutoEval* is in line with how users would judge the results. Here, we are not interested in the *absolute* values but instead we would like to know if the *relative* comparison between different systems can be replicated by user judgements. In other words, we would like to find out whether a query suggestion model deemed better by *AutoEval* is in fact producing "better" query recommendations when consulting real users.

5.3.1. Selected Models

We select the two evolving log-based domain models introduced in Chapter 3 which are continuously learning from past queries as recorded in log files. In addition, we use an association rule-based approach that operates on the same log data. The three models can be summarized as follows:

1. **ACO**: This is the Ant Colony Optimisation model configured as follows:
    - No evaporation; the evaporation co-efficient is set to zero $\rho = 0$.
    - The pheromone calculation scheme is set to 'subsequent reformulations'.
    - Recommendation is done with Depth 1.

2. **Nootropia**: This is the Nootropia model configured as follows:
   - 'Session windows' were used to define context.
   - The type of links is 'Symmetric links'.
   - Recommendation with 'Nodes+Link' weighting scheme.
3. **Association Rules**: This is an alternative to graph-based structures which derives query recommendations using association rules (Fonseca et al., 2003). The idea is to use session boundaries and to treat each session as a transaction. Related queries are derived from queries submitted within the same transaction.

5.3.2. Search Log Data

Again in this experiment we used the University of Essex search logs presented in Section 3.1.7. It contains all the information usually found in typical search logs, i.e. the queries that have been submitted by a user in a single session, the documents displayed to the user and the ones which were clicked. For the experiment we use a portion of the log data between the beginning of the academic year 2008 (01 October 2008) to the end of the autumn term in 2009 (23 December 2009) using the different models for suggesting query modifications. We denote this portion of search log as $\mathscr{L}_{SX08-09}$.

We use weekly batches to split the search log into batches of a continuously updated search log. This resulted in 64 different batches of logs corresponding to 64 weeks. The total number of sessions in this log data is $|\mathscr{S}(\mathscr{L}_{SX08-09})| = 448,025$, with an average of 7000.39 sessions per week. But the number of sessions vary between 2,000 sessions and 13,424 with a standard deviation of $\sigma = 2,142.95$. The variation in number of sessions between batches is due to some busy periods throughout the academic year. For instance, each of the first weeks of both academic years 2008 and 2009 consists of 11,627 sessions and 13,424 sessions respectively,

which are all well above the average. On the other hand, the Christmas week in 2008 contains only 2000 user sessions.

The sessions used for training our models and then testing it using the *AutoEval* approach are only those which contain query reformulations and they make up 29.2% of the sessions. This is lower on that previously reported in Web search log analysis studies. For example, Spink and Jansen (2004) reported a ratio between approximately 42% and 60% for sessions longer than one queries in three different Web Search engines (AllTheWeb, AltaVista, and Excite). Following the approach in (Fonseca et al., 2004) and to reduce noise, in this experiment we only consider sessions where the number of queries is less than 10 and those which span a period of less than 10 minutes. In the end, the number of the sessions used for testing and training was 112,677 (25.14%) and the number of sessions in the batches varied between 516 and 3,379 sessions.

5.3.3. Applying AutoEval

We apply *AutoEval* on the log data $\mathscr{L}_{SX08-09}$ which spans a period of 64 weeks between the beginning of the academic year 2008 to the end of the autumn term in 2009 using the three different models for query recommendations. We have already illustrated how ACO and Nootropia can be updated from a continuously updated search log. We implemented the process of deriving recommendations using association rules using a batch of search logs as described in (Fonseca et al., 2004). The process can be also extended to a continuously updated search log $\mathscr{L}_i$ as association rules can be updated for every new batch using a union of all the previous batches and the current batch $\bigcup_{k=1}^{k=i} \mathscr{L}_k$.

After running *AutoEval* we end up with *MRR* scores for each model on weekly intervals.

5.3.4. User Study

In order to validate our automatic evaluation methodology we performed a user-based assessment as proposed in the literature (Boldi et al., 2009; Sanderson and Croft, 1999). In this approach participants are given sample queries and recommendations as alternatives to those queries and they are asked to determine whether the suggestions are relevant to the original query. In similar work (Boldi et al., 2009), sampling queries for assessment was done by eliminating very frequent or very infrequent queries. However this was in the context of a Web search engine. In our case we are dealing with a local search engine, top frequent queries are important and not necessarily navigational (e.g. timetable in the studied domain is an ambiguous query where users may be interested in finding the timetable of the exams in their department or may want to navigate to the timetable office page to book a room), thus we needed to consider both top frequent queries and queries whose frequencies are in the middle range. In line with (Boldi et al., 2009), we avoided infrequent queries as we assume that these tend to result in no recommendations and we also assume they would not be useful to the user.

We sampled 20 queries for the study as follows. First we randomly selected 10 queries from the top 50 queries in the entire log data up to the point when we conducted the study (15 September 2010). The most frequent submitted query in this data 'library' has a frequency of $f(\text{'library'}) = 27,133$. Then we randomly selected 10 queries within a range of medium frequency (between 50-1000).The sampled set of queries are listed in Table 5.1.

No.	Query Phrase	No.	Query Phrase	No.	Query Phrase	No.	Query Phrase
1	exam timetable	6	accommodation	11	resit	16	biochemistry
2	registry	7	printing credit	12	political science	17	job centre
3	cmr	8	graduation	13	chaplaincy	18	linguistics
4	jobs	9	webmail	14	psychoanalysis	19	course fees
5	enrol	10	courses	15	planning	20	online submission

Table 5.1.: A list of the sampled queries.

In order to select a sensible number of query recommendations, for each sampled query we selected the three best (highest weighted) related queries using five different models:

- ACO_{full}: this is the ant colony optimisation model learnt over the entire 64 weeks period used in the *AutoEval* run.
- ACO_{Aut08}: this is the ant colony optimisation model learnt over a shorter period which is only the autumn term of the academic year 2008.
- *Association Rules*: this is the domain model learnt using Fonseca's association rules over the entire 64 weeks period used in the *AutoEval* run.
- *Nootropia*: this is the domain model learnt using Nootropia over the entire 64 weeks period used in the *AutoEval* run.
- *Baseline*: As a baseline we selected a method that does not rely on log data (and does not get updated in weekly batches). We assume that the top matching results of a commercial search engine will be a useful resource to derive query modification suggestions. We derived nouns and noun phrases from the top ten snippets returned by *Yahoo!* (restricting the search to the University of Essex website). We identify nouns and noun phrases using text processing methods applied in previous experiments (Kruschwitz, 2005).

This has resulted in 214 distinct query pairs after removing duplicates due to the overlap of some suggestions from different systems. An online survey was prepared similar to the one used in (Kruschwitz, 2003), and we asked 16 subjects (students and staff at Essex University) to fill in the survey. Each subject had to judge all the 214 pairs, which means that each pair is judged 16 times. This has resulted in $214 * 6 = 3,424$ pair judgements. Each participant was paid £10 to complete the survey. We could have used a crowd-sourced approach where we collect a large number of pair judgements at a reasonable cost, e.g. by using the popular Amazon Mechanical Turk platform [1]. However, the task is domain specific and only typical users of the search

[1] http://mturk.com

engine (i.e. university staff and students) are capable of providing high-quality judgements that can be useful for our evaluation.

Participants were not told that various different techniques have been used to generate these query pairs. The form contained a list of all query pairs in random order. With each query pair the participants had to decide whether the suggestion was relevant or not relevant. They were also given the choice to choose "do not know" if they were not sure.

The (written) introduction was the following (with a link to the form that contained the selected pairs of related terms):

> *You are the user of a new search engine which searches the University of Essex intranet. In addition to returning the best matching documents for any given query, this search engine also returns a set of words or phrases (called terms)*
>
> - *that give an indication of what the retrieved documents are about, and*
> - *which can be added to the query in order to refine the search or replace the original query*
>
> *The form below gives a list of term pairs. For each pair, imagine the first term was your original query, and that the second is one of the terms proposed by the search system, which you could use to refine or replace the search. Please judge for each pair whether you think the second term is:*
>
> - *relevant (Choose "Yes")*
> - *not relevant (Choose "No")*
> - *If you do not know, then Choose "Don't know".*
>
> *Here, "relevant" means that you can imagine a situation where the second term is an appropriate refinement or replacement of the query given by the first term.*

When considering relevance, remember the particular document collection that is being searched.

5.3.5. Results and Discussion

Figure 5.2 illustrates the results of running *AutoEval*. We see that despite a few spikes the general trend is upwards indicating that different evolving log-based domain models are able to learn from past log data over time. We can also observe relatively low *MRR* scores but we will come to that in the next chapter.

The figure also serves as a comparative tool to assess the performance of different evolving models over time. These models are evaluated and trained on the same data that reflects real user interaction with the search engine. The figure suggests that the ACO method is significantly more effective than learning based on association rules or Nootropia.

As for the *user assessments*, we received a total number of 3,424 assessments for the pairs distributed as in Table 5.2. The ratio of positive judgements is higher that what was reported in a similar study on Web queries (Boldi et al., 2009).

Assessment	Number	Percentage
Relevant	1667	48.69%
Not Relevant	1437	41.97%
Don't Know	320	9.35%

Table 5.2.: Distribution of assessments $n = 3,424$.

We estimated the inter-assessor agreement using two measures, namely the overall percentage agreement P_a and Fleiss' Kappa κ, considering each label of assessment ('Relevant','Not Relevant', 'Don't Know') as a different category. We observe a fair amount of agreement among the assessors ($P_a = 58\%$ and $\kappa = 0.37$), which is comparable to what is reported in (Boldi et al., 2009) ($P_a = 68\%$ and $\kappa = 0.43$). To give the reader a flavour of query pairs that

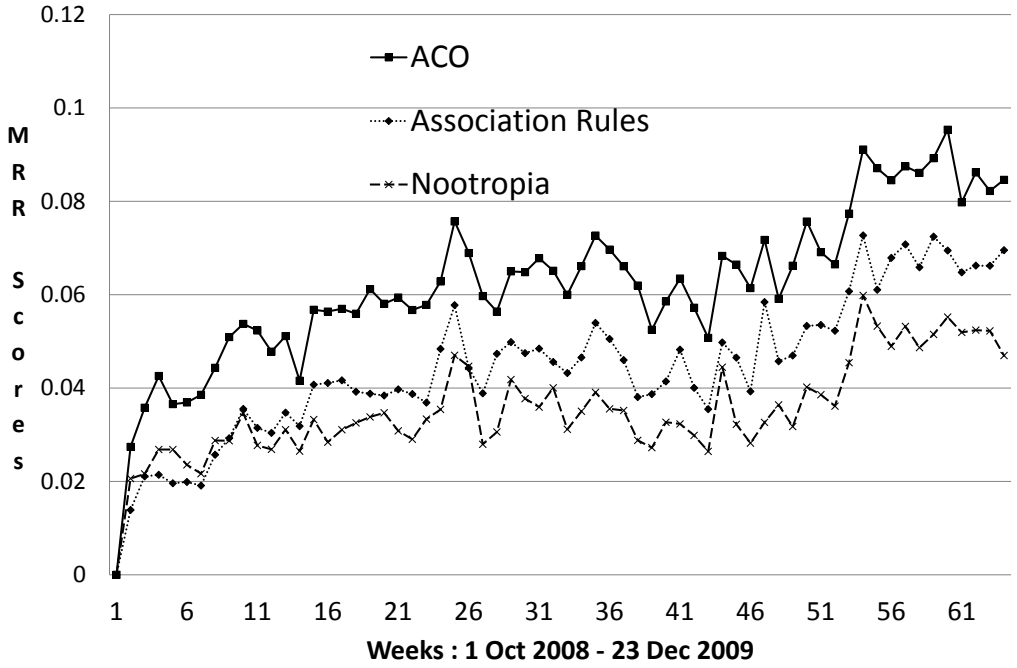


Figure 5.2.: AutoEval run for ACO, Nootropia, and Association Rules for a period of 64 weeks.

received certain assessment, in Table 5.3 we give examples of pairs where the majority of assessors gave the same answer.

Answer	**Examples**
Relevant	'jobs' → 'vacancies', 'printing credit' → 'buy printing'
Not Relevant	'biochemistry' → 'price', 'psychoanalysis' → 'distance'
Don't Know	'planning' → 'sss reports'

Table 5.3.: Examples of pairs that the majority of users gave the same answer to.

The aggregated results of user assessments are shown in Table 5.4. For each assessor we calculated the percentage of pairs that were judged relevant and then we aggregated the results among the assessors. The 'Total Relevant' row gives the average judgement of query pairs considered relevant over all users and all three suggestions for each method. If we only take

into account the top suggestion (i.e. the one highest ranked by the corresponding method) for each query, then we get the results listed under 'First Relevant'. Finally, the percentage for which the system in question had provided at least one suggestion that was judged relevant is shown in the bottom row under 'At Least One Relevant'.

Using the 'Total Relevant' scores, ACO_{full} is the best performing system overall being significantly better than any of the alternatives ($p < 0.05$). The differences in the user assessment scores reflect the differences observed in Figure 5.2. The order of the three different evolving log-based models is consistent with the automatic evaluation. These observations are also true when considering the scores in the 'First Relevant' and 'At Least One Relevant' criterion except on one occasion where the *Association Rules* model outperformed ACO_{full} using the 'First Relevant' aggregated assessment score.

The user assessment also shows that the log-based *ACO* and *Association Rules* models are considered better by the users than the snippet baseline approach which is in line with previous experiments (Dignum et al., 2010).

Furthermore, ACO_{full} is significantly better than ACO_{Aut08} ($p < 0.0001$), i.e. the increase in performance observed over time in the automatic evaluation is reflected in the user assessment. It also means the ACO evolving model is capable of learning better query recommendations over time.

	ACO_{full}	ACO_{Aut08}	*Association Rules*	*Nootropia*	*Baseline*
Total Relevant	59.38% †	50.00%	55.63%	29.16%	54.06%
First Relevant	62.19%	57.12%	64.80%	28.62%	54.42%
At Least One Relevant	84.06% ‡	77.80%	83.44%	61.51%	81.56%

Table 5.4.: Results of user assessments. † means statistical significance (at $p < 0.05$) over all other systems using paired two-tailed t-tests on the scores of individual assessors. ‡ means statistical significance (at $p < 0.05$) over all other systems apart from *Association Rules* using paired two-tailed t-tests on the scores of individual assessors.

5.4. Concluding Remarks

As a conclusion, we have proposed *AutoEval* and shown that it is a sensible methodology capable of identifying performance improvement of an evolving model for providing query suggestions over time. We show that this methodology can perform comparative experiments where different evolving models can be tested under the same experimental conditions.

In the following chapter, we will exploit *AutoEval* to perform a series of evaluations to explore variations of the different models presented in previous chapters.

Chapter 6.

In-Vitro Evaluation of Adaptive Domain Models

In the previous chapters, we have presented a number of methods to build evolving domain models from search logs. However we have not evaluated how *effective* these models are for query recommendation or how *adaptive* they are in real world scenarios. We also did not understand the impact of using different configuration parameters for each individual model and therefore on the quality of recommendations.

In Chapter 5 we proposed *AutoEval*, an evaluation framework that is capable of automatically assessing the performance of query recommendation systems over time based on actual query logs. This framework is capable of automatically measuring the performance of an evolving model to provide query recommendations over time and comparing a number of different models under the same experimental conditions.

In this chapter we exploit this evaluation framework to systematically explore our evolving domain models for query recommendation.

Both the Ant Colony Optimisation (ACO) and Nootropia models are thoroughly tested and their performance is compared with a sensible baseline approach. Also the variations of the enriched query flow graph with click data are evaluated using search logs collected from two academic institutions.

The evaluation framework does not only measure the performance of these algorithms over time but it also helps us in exploring the effects of different parameters on these algorithms which resulted in a deeper understanding of what factors are effective in generating good query recommendations and which ones are not.

We also explore the framework's capability in capturing the effect of seasonality in query recommendation which can be important for the context of intranets or local Web site search. By seasonality we mean that some query suggestions may be more relevant for a certain season than others. For example for the query 'timetable', which is one of the top queries in our studied University search logs, the recommendation 'exam timetable' may be more relevant than 'teaching timetable' in the academic summer term when the actual exams take place. The opposite may be true in the academic autumn term when the academic year starts.

6.1. Evaluation of the ACO Model

We aim to assess the performance of the ACO model presented in Section 3.2 in learning query recommendations over time and study the effect of different parameters on the performance. In particular we would like to address these questions:

- Does the ACO model learn better suggestions over time?
- What are the effects of using direct or indirect associations in ACO?
- What are the effects of different pheromone calculation schemes in ACO?

- What are the effects of different evaporation co-efficient factors in ACO?

6.1.1. Experimental Setup

In this section we first provide a description of the search logs used in these experiments. Then we introduce the experimental design and illustrate the different models being tested.

The *search log data* used in these experiments is our main dataset which is the University of Essex search logs presented in Section 3.1.7. For the ACO experiment we use a portion of the log data which covers a period of three years from (20 November 2007) when the log data was first collected to the same date in 2010. We denote this portion of the search log as $\mathscr{L}_{SX-3yrs}$. We split this search log into weekly batches which has resulted in 155 different batches of logs corresponding to 155 weeks. This is less than the expected 157 batches as the search engine was down for a couple of weeks in August 2008. The total number of sessions in this log data is $|\mathscr{S}(\mathscr{L}_{SX-3yrs})| = 1,040,697$, with an average of 6,714.05 sessions per week. But the number of sessions vary between 1,065 sessions and 14,572 with a standard deviation of $\sigma = 2,457.99$. This is in line with the smaller portion we used before in Section 5.3. The variation in number of sessions between batches is due to some busy periods throughout the year at the University.

The sessions used for training our models and then testing them using the *AutoEval* approach are only those which contain query reformulations and they make up 29.53% of the sessions. Following the approach in (Fonseca et al., 2004) and to reduce noise, in these experiments we only consider sessions where the number of queries is equal or less than 10 and those which span over a period that is not longer than 10 minutes. In the end, the number of the sessions used for testing and training was 264,078 (25.38%) and the batches varied between 365 and 3,948 sessions.

We applied *AutoEval* on the log data $\mathscr{L}_{SX-3yrs}$ using the ACO model to evaluate the performance of the ACO model and the effect of different parameters that are explained in Section 3.2.

The following ACO configurations have been tested:

1. **Depth**: As mentioned before, the depth refers to the recommendation approach. It is the maximum distance away from the original node that is considered to recommend queries. Two values are used (depth = 1, depth =2). The default is 1.

2. **The pheromone calculation scheme**: As mentioned before, three different calculation schemes are considered ('Subsequent reformulations', 'Linking All', 'Link to Last') The default one is 'Subsequent Reformulations'.

3. **The evaporation co-efficient factor**. We use different values to test the effect of the evaporation co-efficient ($\rho = 0, 0.05, 0.1, 0.15, 0.3, 0.5, 0.7, 0.9$). The default value is 0.

AutoEval has been run on a combination of settings for these three parameters. The following combinations have been used: ACO, ACO(depth=2), ACO($\rho = 0.05$), ACO($\rho = 0.1$), ACO($\rho = 0.15$), ACO($\rho = 0.3$), ACO($\rho = 0.5$), ACO($\rho = 0.7$), ACO($\rho = 0.9$), ACO(Link To Last), ACO(Linking All).

Note that when the parameter is not mentioned the default value is used.

6.1.2. AutoEval Results for ACO Variants

After running the evaluation framework for various configurations of the ACO model as described in the previous section, we obtain the weekly *MRR* scores for each model. As described in Chapter 5, for each query pair we calculate the (reciprocal rank) *RR* of the reformulation in the list of the recommendation provided by the model in question. This can vary between 0 (not in the list) and 1 (top of the list). Then we take the mean of these scores

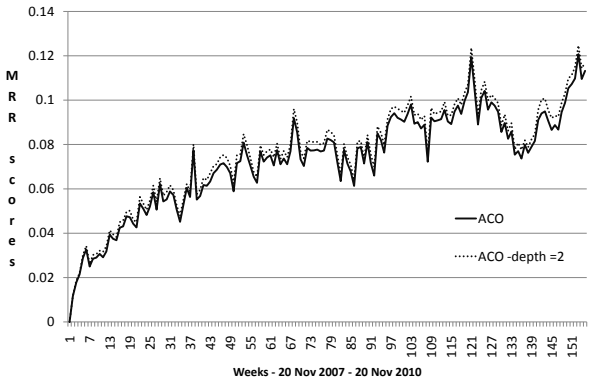


Figure 6.1.: ACO, depth and evaporation factors

for all the pairs in each week. It should be noted that this is different from the traditional notion of *MRR* we scores are averages over distinct queries. Using the time series of the *MRR* scores, we can assess the model's performance over time, and compare the performances of different models.

Table 6.1 lists the average MRR scores across all the weeks of the entire three year period and summarises for all ACO variations the average percentage increase of the *MRR* scores over the default ACO model and the p value of the paired two-tailed t-test against the default ACO model.

Overall, we observe that the obtained *MRR* scores are low. This is possibly due to the *conservative* evaluation where the gold standard is only the reformulation (i.e. the follow-up query) entered by the user. For example if the original query is 'registration' and the follow-up query is 'registration fees', a model that recommend 'fees' will not be rewarded. We will further investigate the low scores when we perform failure analysis with the success rate metric in the next section.

The difference between ACO and ACO(depth=2) is significant with an average percentage increase of 4.35% which suggests that using indirect associations can be useful. This is in line

vs. ACO	Avg. *MRR*(0.0732)	per. increase(%)	paired t-test
ACO(depth = 2)	0.07638	+4.35	< 0.001
ACO(Linking All)	0.07259	-0.64	< 0.001
ACO(Link Last)	0.0649	-15.77	< 0.001
ACO($\rho = 0.05$)	0.0739	+0.25	< 0.05
ACO($\rho = 0.1$)	0.0742	+0.55	< 0.05
ACO($\rho = 0.15$)	0.0741	-0.06	0.20
ACO($\rho = 0.3$)	0.0727	-0.61	< 0.001
ACO($\rho = 0.5$)	0.0714	-2.43	< 0.001
ACO($\rho = 0.7$)	0.0688	-6.09	< 0.001
ACO($\rho = 0.9$)	0.0636	-13.02	< 0.001

Table 6.1.: AutoEval assessment of ACO variations.

with previous findings on recommending queries with the query flow graph model in (Boldi et al., 2008) where better recommendations were achieved by the more advanced random walk and PageRank scoring.

A plotted line graph of the *MRR* scores obtained for the ACO model with variations for the depth parameter is shown in Figure 6.1. With both ACO variations we see that despite the spikes the general trend is upwards indicating that both ACO models are able to learn from past log data over time. We will discuss the spikes later.

In the second section of Table 6.1, the pheromone calculation scheme (building the edges on the graphs) are tested. The 'Subsequent Reformulations' scheme significantly outperforms the other two 'Linking All' and 'Link to Last'. In the 'Linking All' scheme, although the cost function penalises reformulations that do not immediately follow a query, it seems that this is still introducing noise to edges built and the performance degrades and does not improve. The 'Link to Last' scheme suffers the most and the main reason could be that a lot of useful links are lost within the session and especially because our session identification method relies on the simple 10-minute and 10-query rule.

In the third and last section of Table 6.1, we see the impact of varying the evaporation co-efficient factor. The performance gets better with small values of ρ as opposed to no evaporation. In other words forgetting less commonly used and possibly seasonally incorrect reformulations has a positive impact. The observed increase on the performance is quite small but statistically significant. However with larger values of the evaporation co-efficient the performance degrades significantly. At $\rho = 0.3$ the performance degrades significantly and the downward trend continues with larger values of ρ. A possible explanation here is that a model that forgets quickly could miss out frequent and time insensitive links in favour of less prominent but more recent and currently trendy links.

6.1.3. A Closer Look at ACO with Evaporation

As we saw in the previous section, the optimum value found for the evaporation factor is 0.1 but the actual increase over ACO, although statistically significant, is quite small. In Table 6.2, we perform failure analysis by measuring the average scores of the success rates at various ranks (3, 5, 10). We see that the ACO(ρ=0.1) is always better but the increase is quite small.

To have a better understanding of how these two differ, we examined all the query reformulations in one of the log batches (batch 139 in the week between July 10th and July 17th 2010) where the both models were evaluated on. We examined the ranks of the actual reformulation (the follow-up query that the user has reformulated his original query into) predicted by each if any. In this batch, the total number of query reformulations was 2,462 query pairs. The scores obtained for both models in this batch are listed in Table 6.3. These scores are in line with the average scores except for *SR*@5. We also report the success rate at any rank *SR* which is the same for both models. The *SR* will always be the same as the direct links of each node in the graph will be identical; only weights (pheromone trails) differ. The *SR* itself is not a good measure as users of traditional search engine interfaces will only be able to cope with few recommendations (e.g. 10). The assumption is that we do not want to overload the

user with a large number of recommendations. In fact in, Web search engines, it has been shown that it is a good practice to limit the number of search results to 10 results per page and that including more results per page has a negative effect Hearst (2009). Similarly, for query recommendations we assume that we only present the top 10 recommendations to the user . Therefore a better performance will be achieved by a better ranking of the links. The *SR* score indicates that for only around 17% of reformulations both models were able to provide the correct suggestion at all in their recommendation list. In other words the two models are competing in only 424 query pairs. One of the reasons for this low ratio and the low *MRR* is the conservative evaluation as discussed in the previous section.

	ACO	ACO($\rho = 0.1$)
MRR	0.0732	0.0742
SR@3	0.0830	0.0842
SR@5	0.0964	0.0981
SR@10	0.1128	0.1137

Table 6.2.: A closer look at ACO.

	ACO	ACO($\rho = 0.1$)
MRR	0.0797	0.0799
SR@3	0.0881	0.0889
SR@5	0.1040	0.1036
SR@10	0.1194	0.1198
SR	0.1722	0.1722

Table 6.3.: Scores on batch 139.

We report some of the examples (pairs) where ACO and ACO(ρ=0.1) have a different ranking r_i of the actual follow-up query in Table 6.4. In the first section of the table we see pairs where the ACO with evaporation provided a better ranking than ACO without evaporation. The week of this batch is the one that precedes the graduation ceremony and we see better ranking for suggestions about graduation queries in ACO with evaporation. Also we see that the model

is better in serving information needs which have a temporal dimension e.g. (term dates 2010) or (fees 2010 2011). In some cases though it was not clear why this model is performing better.

In the second section of the table, we see example were the ACO without evaporation was superior. These queries tend to represent information needs which are either general, less context-sensitive or less season-sensitive.

Query Pair	$RR_{ACO(\rho=0.1)}$	RR_{ACO}
computer→computer help desk	**0.250**	0.200
graduation→graduation photos	**0.090**	0.083
term dates→term dates 2010	**0.067**	0.059
msc→msc courses	**0.250**	0.200
mba→fees	**0.143**	0.125
fees→fees 2010 2011	**0.077**	0.067
summer school→international academy	**0.334**	0.250
fees→mba fees	**0.091**	0.083
registry→graduation office	**0.100**	0.091
graduation→graduation dates	**0.083**	0.077
summer school→taster courses	**0.021**	0.02
music→sound	**0.083**	0.077
telephone→internal telephone	**0.333**	0.250
dissertation→ma dissertation	0.500	**1.000**
courses→current courses	0.167	**0.200**
estates→estates office	0.047	**0.050**
graphic design→design	0.200	**0.250**
map→site map	0.077	**0.100**
english→ba english	0.038	**0.040**

Table 6.4.: Query pair examples from batch 139 and the reciprocal rank of the modification in both ACO(ρ=0.1) and ACO.

batch	108 ↓	121 ↑
$\Delta(MRR)$	-0.0166	+0.0155
Number of pairs	1,037	4,097
Frequent Queries %	9.4%	22.7%

Table 6.5.: Statistics from the batches of the spikes.

6.1.4. Performance Consistency

As we see in Figure 6.1, there is an upward trend of the *MRR* scores over time and this is also true for all of the ACO variations.

To measure this upward trend, we take the ACO($\rho = 0.1$) model as an example and perform a standard linear regression between time (the batch number i) and the observed *MRR* score of the model at a certain batch. The regression yields a linear relationship specified with Equation 6.1.

$$MRR = 0.0004 * i + 0.0385 \tag{6.1}$$

This linear relationship is statistically significant using a t-test with a confidence of $p \ll 0.001$.

Despite this upward trend we see that there are also spikes in the line plotted for the *MRR* scores of the model in Figure 6.1. We examined two spikes to understand this behaviour; an upward spike and a downward spike. We report some statistics about the two batches corresponding to the two spikes in Table 6.5.

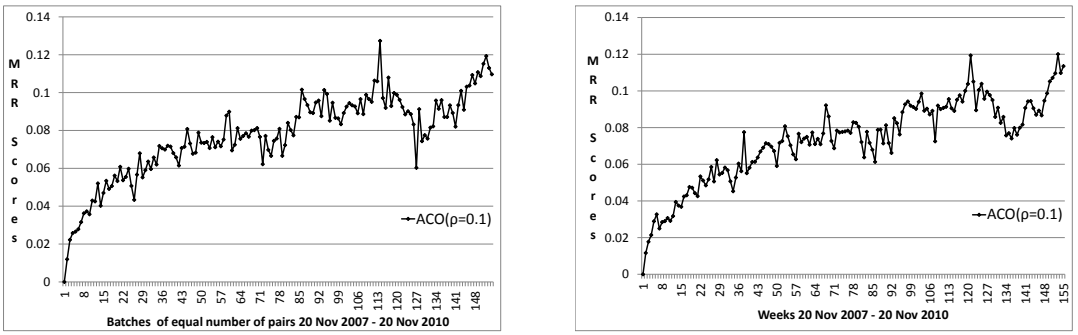
We see that there is a big gap in the number of the pairs in those two batches and both are far from the average number of pairs in all the batches which is 2,820.

Another interesting observation is the percentage of the 10 most frequent queries in the entire 3 year log found in the pairs. By this percentage we mean the ratio of queries in the pairs which contains a very frequent query. In the upward spike we see a high ratio of those.

When we examine the query pairs in the batches, we observe that in the upward spike there are a lot of identical pairs where the model is performing well. For example the pair (timetable $\rightarrow$ exam timetable) was observed 30 times in the batch which is not surprising as the batch corresponds to a week during the examination period. The *RR* score of the model for this pair is 0.5. Moreover the ratio of queries in the pairs which contain the query "timetable" is 16.7%.

On the other hand, in the downward spike we observe a number of noisy queries, possibly test queries which were not captured by the 10-query and 10-minute limit of our session identification method.

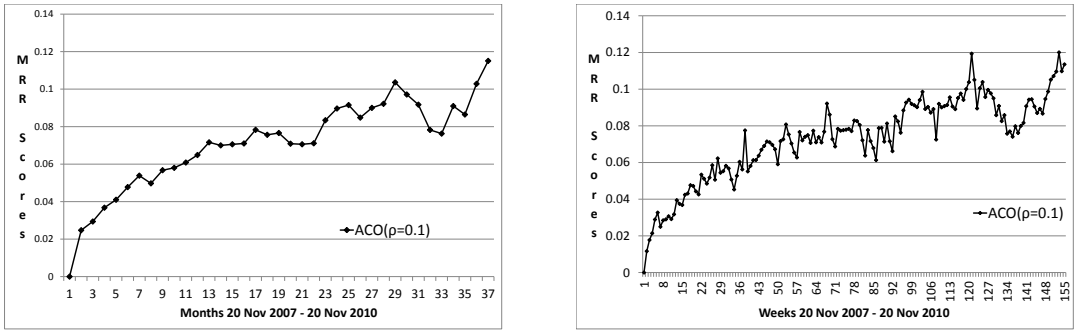
To eliminate the variable number of query pairs in a single batch (a week), we divided the entire search log into the same number of batches but with equal number of query reformulations (query pairs) as identified before in their chronological order. Thus the number of query pairs was selected to be the average number of pairs in all the weekly batches used before. We see a similar shape of the plot of *MRR* score, Figure 6.2, with spikes up and down.



(a) Equal batches **(b)** Weekly batches

Figure 6.2.: Comparison between AutoEval runs for ACO($\rho = 0.1$).

Also, we did an experiment where we divided the search log into monthly batches instead. In Figure 6.3 we plot the monthly *MRR* scores and we see a more consistent and smoother increase of the performance.



(a) Monthly batches **(b)** Weekly batches

Figure 6.3.: Comparison between AutoEval runs for ACO($\rho = 0.1$).

Finally, we can observe from Figure 6.2, that in fact the ACO model becomes better with more data. The performance of ACO in the second year (the average *MRR* score) is better than the first year. However, the performance in the third year is only slightly better than the third year, which means the performance becomes more stable. It will be interesting to be able to quantify how much data is needed for training to make ACO stable but we leave that for future work.

6.2. Evaluation of the Nootropia Model

In Section 3.3 we presented how the immune-based model originally created for Information Filtering, Nootropia, was applied to build an adaptive domain model from search logs. We also demonstrated how to use that model for query recommendation.

The model as presented could be configured in various ways. We aim to assess the performance of the Nootropia model in learning query recommendations over time and study the effect of different configurations on the performance. In particular we would like to address these questions:

- Does the Nootropia model learn better suggestions over time?

- How do symmetric links compare to non-symmetric links?
- What are the effects of different weighting schemes in Nootropia?
- What are the effects of different context definition window in Nootropia?

6.2.1. Experimental Setup

Due to limited computational power in the lab we decided to run the Nootropia experiments over a shorter but still substantial portion of the University of Essex search logs $\mathscr{L}_{SX08sh}$. This portion covers the period from (1 January 2008) to (15 March 2008). As in the previous experiment we split this search log into weekly batches and we only consider sessions where the number of queries is equal or less than 10 and those which span over a period that is not longer than 10 minutes.

The experiments apply *AutoEval* using the Nootropia model to evaluate the performance of Nootropia in providing query recommendations over time and the effect of different parameters on that performance. The Nootropia configurations explained in Section 3.3 are tested. This includes (i) Symmetric and Non-Symmetric links, (ii) the window size and (iii) the weighting schemes for producing recommendations. We first run a series of experiments with symmetric links and session windows, combined with the three weighting schemes ('nodes' weighting, 'links' weighting and 'nodes+links' weighting). We then evaluate the best performing weighting scheme combined with non-symmetric links and session windows. Finally, to evaluate the effect of context we performed a final experimental run using pair windows.

6.2.2. Nootropia Results

In Figure 6.4a we plot the *MRR* scores for the Nootropia Symmetric model using different weighting schemes.

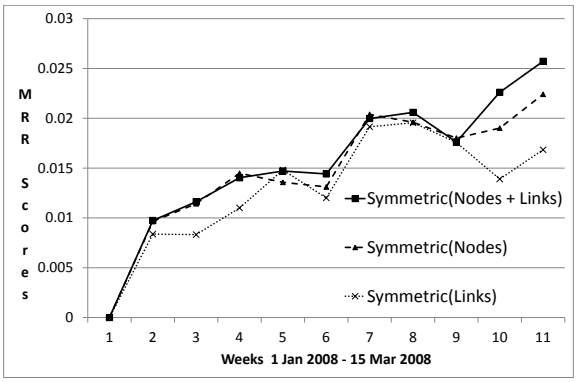
Link Type	Window Size	Weighting	MRR
Symmetric	Session	Links	0.0129
		Nodes	0.0146
		Links + Nodes	0.0155 †
Non-Symmetric	Session	Links + Nodes	0.0161 ⋄
	Pair	Links + Nodes	**0.0205** ‡

Table 6.6.: Summary of the results for Nootropia. † means statistically significant increase over both the 'Links' and the 'Nodes' weighting using paired 2-tailed t-test (at $p < 0.05$) and average percentage increase of +22.80% and +5.25% respectively. ⋄ means an improvement over the 'Symmetric' model with average percentage increase of +1.73%. ‡ means statistically significant increase over the 'Session' windowing (at $p < 0.01$) with average percentage increase of 28.06%.

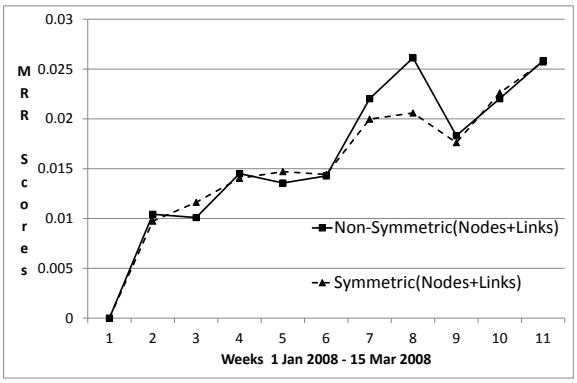
The first three lines of Table 6.6 summarise the results for the 'Symmetric' Nootropia model. Note that session windows were used here. According to the results for the 'Symmetric' model, node weights produce better query recommendations than link weights alone. However the model works better when combining both of them. This indicates that both query occurrence and co-occurrence statistics provide useful data when building a domain model for query recommendation.

In line 4 of Table 6.6 we see that the average *MRR* scores for 'Non-Symmetric' model are better than the 'Symmetric' one. The comparison of the plots are illustrated in Figure 6.4b. This finding shows that the sequence of queries in a query session is an important aspect of the process that should be taken into account. Sessions should not be treated as "bag of queries".

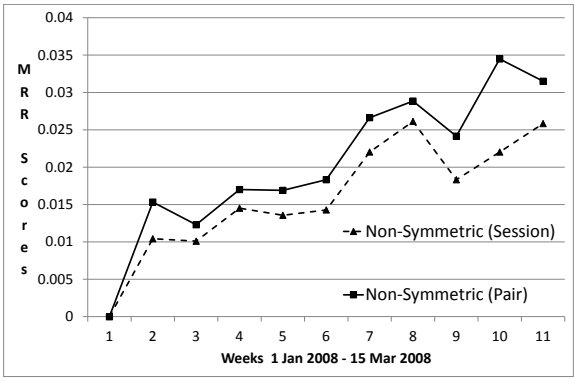
Using the last two lines in Table 6.6 we can compare the variation of the window size using the 'Non-Symmetric' model and Figure 6.4c plots the *MRR* scores for those models. The best results for Nootropia are achieved when pair windows are adopted. This indicates that linking all queries in a session introduces additional noise which is line with previous findings when we tested the different pheromone calculation schemes of the ACO model (see Section 6.1.2). In contrast, with the pair window a query modification is only linked to the initial query that caused this modification.



(a) Nootropia 'Symmetric' with 'Session' windows



(b) Nootropia with 'Session' windows and 'Node+Links' weighting



(c) Nootropia 'Non-Symmetric' with weighting of 'Nodes+Links'

Figure 6.4.: Nootropia intra-model comparisons of the plotted MRR scores.

6.3. Inter-model Comparison

We use *AutoEval* to compare both ACO and Nootropia models against a simple alternative based on association rules (Fonseca et al., 2003). We have already introduced the 'Association Rules' model in Section 5.3 This approach represents a sensible baseline for a different way of providing adaptive recommendations because it accesses exactly the same resources as our proposed methods and it has been shown to work well on product search (Fonseca et al., 2003).

	per. increase(%)	**paired t-test**
ACO($\rho = 0.1$) vs. Association Rules	248.71	< 0.001
Nootropia Best vs. Association Rules	208.14	< 0.001
ACO($\rho = 0.1$) vs. Nootropia Best	18.27	< 0.001

Table 6.7.: Summary of the results for inter-model comparison.

Figure 6.5 illustrates the results of running the automatic evaluation framework to perform a comparison between our adaptive approaches and the 'Association Rules' baseline. We used the log data $\mathscr{L}_{SX08sh}$, the one used in the previous section for evaluating the Nootropia model. Note that the system 'Nootropia Best' is the one which we identified from the results in the previous section. The best performing Nootropia variation is a non-symmetric model with 'Nodes+Links' weighting scheme and trained on pair windows. Both ACO and Nootropia models are outperforming the baseline. This demonstrates the power of using an automatic evaluation framework to experiment in-vitro with various models based on real-world data and improve their performance through appropriate modifications. We are improving our models through experimentation and both models were better than the baseline.

The results also show that ACO is better than Nootropia and this is clearly due to the quality of links that it produces (since there are no node weights in the case of ACO).

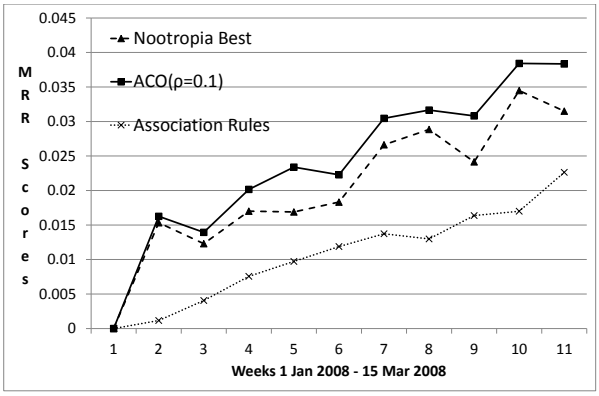


Figure 6.5.: Comparing ACO and Nootropia against a baseline

6.4. The ACO Model and Seasonality Factors in Query Recommendation

As we have seen, one of the motivating points for the ACO analogy is the ability of the model to automatically adapt to new trends and reinforce them and gradually reduce the importance of associations which are becoming less popular or unfashionable via the pheromone evaporation process. In the context of our query recommendation model, the model should be able to recommend seasonally relevant queries as a side effect of the evaporation process. However, this might not always be the case due to the sparsity of the dataset.

In a university context, seasonal divisions of the information needs of the user population during an academic year may possibly be mapped to the academic terms at the university. The academic year at the university at hand is divided into three academic terms (Autumn, Spring, Summer). Each academic term consists of 10 working weeks and they are the most active weeks of the year in which students attend lectures or do exams. The exact dates of these academic terms differ slightly from year to year.

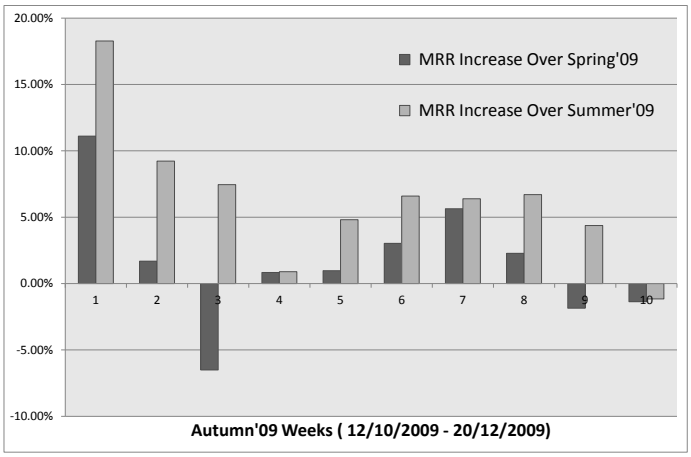
In this section, we want to test whether an ACO model built from the logs of a particular academic term (season) works better than an ACO model built from a period of a similar length,

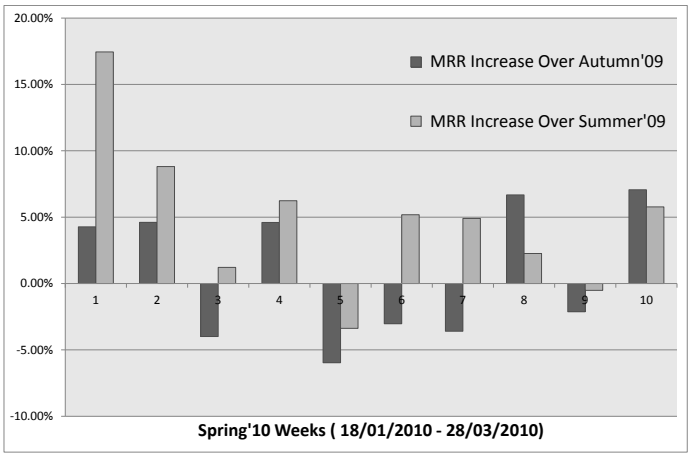
that spans over other parts of the academic year, when both models are evaluated against that particular academic term in the following year.

We did a *cross-season evaluation* between the academic terms of the academic year 2008/2009 and the academic year 2009/2010. Under this evaluation we train the ACO model on a certain academic term in the first academic year and then perform an *incremental AutoEval* on each of the academic terms of the following academic year. By *incremental AutoEval* we mean a continuous cycle of testing then learning as before. In other words we train an ACO model on the academic Autumn term of 2008/2009, then we evaluate it on each of the academic terms of 2009/2010. We repeat this process for the Spring and Summer academic terms of 2008/2009. At the end, we can compare the performance of the three models on each term of the second academic year.

In Table 6.8, we report the results of running the cross-season evaluation. Overall, we see that models trained on the same academic term (season) of a previous year outperform models trained on different academic terms (seasons). The average *MRR* scores of a same-season model (i.e. a model that is trained then tested on the same academic term) are higher than the other two for each testing period, i.e. each academic term of the academic year 2009/2010. Although in some cases it is only slightly higher.

We also report the percentage increase/decrease on the *MRR* scores of the same-season models over the other two for each academic term in Figure 6.6. We see that positive differences are dominant across the three testing periods. We also see that the differences are more significant when the academic term in question (training period of the model we compare the same-season model against) is further away from the starting day of the testing period. For example, for Autumn'09 the differences of the *MRR* scores of the model trained on Autumn'08 over the *MRR* scores of the model trained on Spring'09 are more significant than the differences over the model trained on Summer'09.





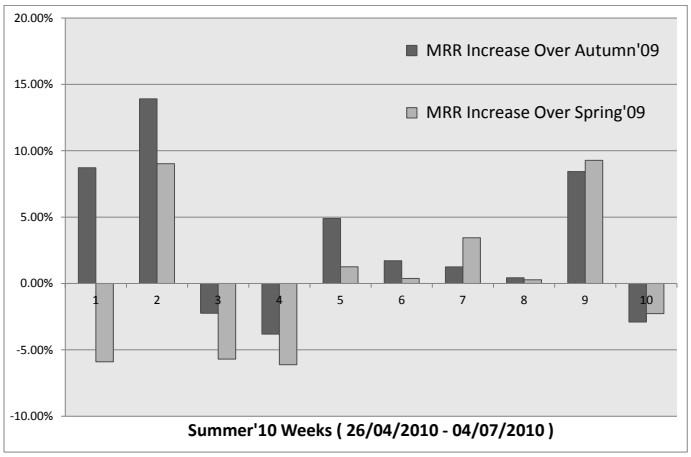


Figure 6.6.: Cross-season evaluation results.

Scores on Autumn'09		
Aut'08	**vs. Spr'09**	**vs. Sum'09**
MRR(0.0628)	0.0619	0.0590
increase(%)	+1.58	+6.36
paired t-test	0.24	< 0.01
Scores on Spring'10		
Spr'09	**vs. Aut'08**	**vs. Sum'09**
MRR(0.0629)	0.0622	0.0600
increase(%)	+0.85	+4.79
paired t-test	0.50	< 0.05
Scores on Summer'10		
Sum'09	**vs. Aut'08**	**vs. Spr'09**
MRR(0.0499)	0.0485	0.0497
increase(%)	+3.04	+0.37
paired t-test	0.15	0.80

Table 6.8.: Cross-season evaluation results.

These results show that there is an element of seasonality that should be taken into account when suggesting queries in a university domain, as models learnt on similar periods were better in general.

However, the evaluation also suggests that defining temporal boundaries for seasonality of information needs can be challenging. Even in a university domain, where there is a regular periodic change in function and activities throughout the year (well defined by academic terms), it was hard to achieve significant improvement in the performance of the recommender model when trained over the same season. For example, using paired two tailed t-test no significant improvement was obtained between Spring'09 and Autumn'08 ($p = 0.50$).

6.5. Evaluation of Enriched Query Flow Graphs

In Chapter 4 we presented an extension of a state-of-the-art model to derive query recommendations from the logs by incorporating click data. We also implemented a variation of the model on our main dataset and illustrated some examples of recommendations generated for sample queries using four different interpretations of the number of clicks upon submitting a reformulation (Section 4.2). However, it is hard to assess the impact of each interpretation for query recommendation. Here we aim to use our automatic evaluation framework *AutoEval* to test those graphs under the same condition on two different search logs, the University of Essex Search logs, and for the first time in this thesis we use another dataset which is the Open University Search Logs.

6.5.1. Experimental Setup

In Section 4.2.1, we did an analysis of the distribution of the number of clicks for each record in the log on a portion of the University of Essex search logs $\mathscr{L}_{SX11}$ (Figure 4.1). Based on this distribution and the finding that less than 2% of all queries result in more than 2 clicks we simplified the general method of producing weights on the edges of the enriched query flow graph into a simpler version, i.e. we simplified Equation 4.2 to Equation 4.5 where we treated all the cases of 2 clicks or more as one case.

Here we want to conduct experiments with *AutoEval* on our main dataset and on search logs of another academic institution, the Open University (OU), where the same sort of data can be obtained (queries, sessions, click data, etc.). Figure 6.7 shows the corresponding histogram for the logs of the OU search engine using exactly the same 10-week period. It has a similar shape to the one of the $\mathscr{L}_{SX11}$ presented in Figure 4.1 with much higher values of counts. In both histograms, for most cases the users either click on one result or do not click on any. Therefore,

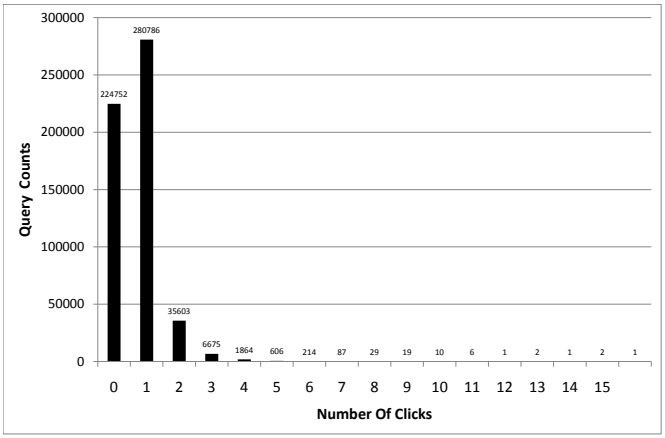


Figure 6.7.: Frequency of queries for each click counts band - OU Search Engine.

it is also reasonable to use the simplified Equation 4.5 to build enriched query flow graphs with this dataset.

In Table 6.9 we again list all the combinations of the frequency co-efficient factors we considered to derive different interpretations of click counts which were already introduced in Section 4.2.2. In this experiment we will run *AutoEval* on both datasets using the 5 different query flow graphs in Table 6.9.

We slightly change the *AutoEval* procedure in this experiment. As discussed in Section 4.1.3 , producing query recommendations from the graph is computationally expensive. Due to computing limitations, when calculating the *MRR* score we consider only a sample of the query reformulations in the batch by taking every tenth query modification.

6.5.2. Evaluation on the University of Essex Search Logs

We applied *AutoEval* on the search log data $\mathscr{L}_{SX11}$ and used weekly batches to calculate the *MRR* scores for each graph.

	C_0	C_1	C_k
$QFG_{standard}$	1.0	1.0	1.0
QFG_{no_zero}	0.0	1.0	1.0
QFG_{boost_one}	1.0	2.0	1.0
$QFG_{boost\_one\_more}$	1.0	3.0	1.0
$QFG_{penalise\_many}$	1.0	2.0	0.5

Table 6.9.: Experimental graphs.

Using the MRR scores, we can assess the graph performance over time in generating query recommendations and compare the performance of different graphs.

Graph	Avg. Weekly Score
QFG_{boost_one}	0.0820
$QFG_{boost\_one\_more}$	0.0817
$QFG_{penalise\_many}$	0.0812
$QFG_{standard}$	0.0789
QFG_{no_zero}	0.0533

Table 6.10.: Average Weekly *MRR* scores obtained for the query flow graphs in UOE search logs. The graphs are ordered by their scores.

Table 6.10 presents the average weekly *MRR* scores obtained (ordered by average score). We observe that the enriched query flow graphs are outperforming the standard query flow graph. Apart from QFG_{no_zero} all enriched graphs are producing higher average *MRR* scores. To perform a statistical analysis on the differences between the enriched query flow graphs, in Table 6.11 we compare the query flow graphs using the average percentage increase of *MRR* scores and the p value of a paired two-tailed t-test.

We observe that when boosting the co-efficient factor of single clicks, statistically significant improvements are obtained. Both QFG_{boost_one} and $QFG_{boost\_one\_more}$ are significantly better than the standard query flow graph $QFG_{standard}$. However no further improvement can be

	per. increase(%)	paired t-test
QFG_{boost_one} vs. $QFG_{standard}$	2.3%	< 0.05
$QFG_{boost\_one\_more}$ vs. $QFG_{standard}$	2.2%	< 0.05
$QFG_{boost\_one\_more}$ vs. QFG_{boost_one}	-0.1%	0.91
$QFG_{penalise\_many}$ vs. QFG_{boost_one}	-0.8%	0.16
QFG_{no_zero} vs. $QFG_{standard}$	-61.2%	< 0.01

Table 6.11.: Comparison of the query flow graphs (UOE search engine).

observed when we further boost the co-efficient factor of single clicks. In fact $QFG_{boost\_one\_more}$ is slightly worse than QFG_{boost_one}.

Comparing $QFG_{penalise\_many}$ to QFG_{boost_one} would inform us about the impact of reducing the co-efficient factor of more than one-click counts. The results show that this does not have a positive impact on the quality of recommendations. $QFG_{penalise\_many}$ is worse than QFG_{boost_one}.

Only enriched graph QFG_{no_zero} failed to improve the *MRR* scores, and in fact it was significantly worse than the standard graph with a high average percentage decrease. This appears to be counter-intuitive as we would assume that queries resulting in no clicks are not good candidates for query recommendation suggestions, and this finding warrants further analysis in future experiments.

In any case, this last finding suggests that completely eliminating reformulations with no user clicks affects the query recommendation quality negatively. Note that in QFG_{boost_one} and $QFG_{boost\_one\_more}$ we are considering these reformulations but we are also penalising them as they have a smaller co-efficient factor.

6.5.3. Evaluation on the Open University Search Logs

To validate the findings we obtained the log files of another academic search engine. To get a comparable number of interactions we decided to run this experiment in daily batches over 10

days of the April 2011 logs, i.e. we now use daily batches to update the graph and calculate the *MRR* scores.

Table 6.12 presents the results obtained in this experiment. The corresponding t-test results can be found in Table 6.13.

Graph	Avg. Daily Score
QFG_{boost_one}	0.0488
$QFG_{penalise\_many}$	0.0480
$QFG_{boost\_one\_more}$	0.0478
$QFG_{standard}$	0.0476
QFG_{no_zero}	0.0425

Table 6.12.: Average Daily *MRR* scores obtained for the query flow graphs in OU search logs. The graphs are ordered by their scores.

	per. increase(%)	paired t-test
QFG_{boost_one} vs. $QFG_{standard}$	2.1%	0.15
$QFG_{boost\_one\_more}$ vs. $QFG_{standard}$	0.1%	0.88
$QFG_{boost\_one\_more}$ vs. QFG_{boost_one}	-2.0%	< 0.05
$QFG_{penalise\_many}$ vs. QFG_{boost_one}	-1.4%	< 0.05
QFG_{no_zero} vs. $QFG_{standard}$	-9.9%	< 0.01

Table 6.13.: Comparison of the query flow graphs (OU search engine).

Despite some minor differences we can see the same pattern. The ordering of the graphs according to their average *MRR* scores is similar. Only positions 2 and 3 ($QFG_{boost\_one\_more}$ and $QFG_{penalise\_many}$) are swapped. The enriched query flow graphs are outperforming the standard query flow graph but no statistical significance was observed this time.

Like before, reducing the co-efficient factor of many click counts did not have a positive impact on this dataset either. In fact $QFG_{penalise\_many}$ is now significantly worse than QFG_{boost_one}. Again, we find that eliminating queries that result in no clicks does not improve performance but instead results are significantly worse.

6.6. Conclusions

In this Chapter, we applied a methodology for the automatic, in-vitro, evaluation of adaptive query recommendation models based on real world data. The methodology is controlled, deterministic and fully reproducible and allowed us to conduct a series of experiments to evaluate variations of two biologically inspired models, ACO and Nootropia, and compare them to a baseline model. We also explored variations of interpreting the post query browsing behaviour in the form of the number of clicked documents to build enriched query flow graphs for query recommendation.

6.6.1. Summary on ACO and Nootropia

Our in-vitro experiments allowed us to quantitatively answer a series of research questions and to draw very useful conclusions. When evaluating the ACO model on search logs of the Web site of our academic organisation, we observed that taking into account indirect associations between queries has a positive effect on the accuracy of the recommendations. This is in line with previous findings that advanced recommendation approaches such as random walks (Boldi et al., 2008) and projections on the query flow graphs (Bordino et al., 2010) perform better. We also show that the evaporation of pheromone trails (forgetting) over time can have a positive effect on the quality of query recommendations but it can also harm performance when forgetting is done on a fast pace i.e. a higher evaporation co-efficient factor. When we examined the model with a customised evaporation co-efficient factor we saw that more seasonally relevant suggestions were provided as a side effect of the evaporation process. Furthermore, our cross-season evaluation results indicate that season-relevant recommendations are useful in the context of a university Web site. However, it was hard to define hard boundaries between seasons.

The evaluation of Nootropia showed that both occurrence and co-occurrence statistics can play an important role when constructing models for query recommendation. Nootropia's results also demonstrate the importance of context and sequence when learning from query log data. This is also in line with the results of evaluating the ACO linking schemes.

Both ACO and Nootropia perform significantly better than the baseline model, Association Rules. This exemplifies the importance of continuous learning for the task at hand. We are confident, that our experimental framework will allow us to further improve our models and learning algorithms and to try further alternatives. According to the overall comparison, ACO is the best performing approach, despite being based only on weighted links between queries. Nootropia's performance is better than the baseline but worse than ACO. This is promising, especially if we consider that it is a model transferred from a quite different application domain. Interestingly, and unlike ACO, its recommendation accuracy is mainly due to the weight of queries (nodes), rather than the weight of links.

6.6.2. Summary on Enriched Query Flow Graphs

Boosting queries which result in a single document click has a positive impact on query recommendation. A single click can be interpreted as quickly reaching a landing page and rewarding these queries significantly improved the automatic evaluation scores. This is line with previous findings on using landing pages to generate query recommendations (Cucerzan and White, 2007).

Eliminating queries which result in no clicks negatively impacted query recommendation. One possible explanation (but certainly only one single aspect) could be that some users found what they are looking for in the result snippets and as a result they would not continue clicking on the right document. Therefore, the graph will miss those useful suggestions. Penalising

these reformulations without completely eliminating them though would have a positive effect as graphs QFG_{boost_one}, $QFG_{boost\_one\_more}$ have a smaller co-efficient factors for zero clicks.

We also show the observation made on one dataset was similar on a different dataset. The performance of the experimented graphs was similar on both datasets. However no statistical significance was observed for the enriched query flow graph over the standard query flow graph on the OU search engine. This may be due to the higher sparsity of the OU search engine logs.

Chapter 7.

Adaptive Models for Query Session Retrieval

In the previous chapters, we have presented a number of approaches for turning search logs into evolving domain models that can adapt to the search trends of the user population. We focused on evaluating these models for query recommendation as an interactive feature that *explicitly* provides the user with the right terminology for her information needs. With a reference to our second research question in Section 1.2, we have only applied our adaptive domain models for only a single IR task of query recommendation. We envisage that these adaptive domain models can also be useful for other IR tasks. We aim to show that these models can be employed in query session retrieval. Query session retrieval extends the traditional ad-hoc retrieval by taking into account previous user interactions with the retrieval system within the same session when answering a query.

In this chapter, we illustrate how log-based domain models can be applied to the problem of session retrieval. We use the Session track of the Text REtrieval Conference (TREC) as our test-bed to evaluate a number of models built from different resources. One of the motivations for

applying these models in this context is the desire to test them in an IR competition environment such that they can be compared with a wide range of alternatives.

The key idea in our approach is to derive query expansions related to the user session from log-based models i.e. *implicitly* supporting the user by mapping her session to similar sessions inferred from the model.

The rest of the chapter reads as follows. First we introduce the tasks in the Session Track and the experimental setup we use. Next we explain and discuss, with experimental evidence, a number of different approaches for tackling the task. Finally we give a summary of the findings.

7.1. The Session Track

Since its launch in 1992, the annual TREC workshop has produced a number of large test collections that fostered IR research in a number of IR problems (Vorhees and Harman, 2005).

The Session Track was introduced at TREC 2010 (Kanoulas et al., 2010). The Session Track tries to evaluate the effectiveness of search engines in interpreting query reformulations. It aims to evaluate the ability of search engines to utilise previous user interactions in order to provide better results for subsequent queries in a user session and therefore 'point way' to what the user is actually looking for.

The goals set for the Session Track are:

- **(G1)** to test whether systems can improve their performance for a given query by using previous user interactions with the retrieval system within the session, and
- **(G2)** to evaluate system performance over an entire query session instead of a single query.

At the time of producing this thesis, the Session Track has run for two consecutive years and the author has lead the 'University of Essex' group submissions which effectively composes the methods we are proposing in this chapter for the session retrieval problem.

7.1.1. The Session Track 2010

In the first year of its launch, the Session Track 2010 focused on sessions consisting of only two queries and further interaction data within the session was *not* provided. In fact, the pairs were simulated from the TREC 2009 Web track 2009 diversity topics (Kanoulas et al., 2010). Participants were given a set of 150 query pairs, each query pair (original query, query reformulation) represents a user session.

The participants were asked to submit three ranked lists of documents from the ClueWeb09 dataset:

- One for the original query ($RL1$).
- One for the query reformulation ignoring the original query ($RL2$).
- One for the query reformulation taking the original query into consideration ($RL3$).

By using the ranked lists ($RL2$) and ($RL3$) we will evaluate the ability of systems to utilize prior history (G1). By using the returned ranked lists ($RL1$) and ($RL3$) we will evaluate the quality of ranking function over the entire session (G2). Based on previous work in analysing query logs, the Session Track identifies three different types of query reformulations. Each query pair provided to participants is considered to belong to one of these types. The session types as explained in (Kanoulas et al., 2010) are:

1. **Generalisation**: In this case, the user starts with a query and gets back some results which may be too narrow or she realises that she wanted a broader spectrum of results, so she reformulates to a more general query, e.g. 'low carb high fat diet' $\rightarrow$ 'types of diets'.

2. **Specification**: In this case, the user starts with a query and gets back some results which may be too broad or she realises that she wanted results within a specific category or subtopic, so she reformulates to a more specific query, e.g. 'us map' → 'us map states and capitals'

3. **Drifting/Parallel Reformulation**: This type represents the case of starting with a query and then reformulating with another query at the same level of specification but with a different aspect of information need, e.g. 'music man performances' → 'music man script'.

The type of query reformulation is not known to the participants. In our submissions as participants we did not attempt to automatically classify a query pair into one of the three categories and therefore we treated all pairs equally.

7.1.2. The Session Track 2011

The main difference in the 2011 task compared to the previous year is that more interactive data is provided to the participants. Moreover the sessions studied here were actually real user sessions collected by the organisers (Kanoulas et al., 2012).

Participants have been provided with a set of query sessions, 76 sessions in total. Each session consists of the current query q_m and the query session prior to the current query:

(a) the set of past queries in the session q_1, q_2, ..., q_{m-1}.

(b) the ranked list of URLs for each past query,

(c) the set of clicked URLs/snippets and the time spent by the user reading the corresponding to each clicked Web page.

Using the ClueWeb09 dataset, participants then run their retrieval system over the current query,

- ignoring the session prior to this query (*RL*1).
- considering only the item (a) above, i.e. the queries prior to the current query (*RL*2).
- considering only the items (a) and (b) above, i.e. the queries prior to the current along with the ranked lists of URLs and the corresponding web pages (*RL*3).
- considering only the items (a), (b) and (c) above, i.e the queries prior to the current, the ranked lists of URLs and the corresponding Web pages as well as the clicked URLs and the time spent on the corresponding Web pages (*RL*4).

By comparing the retrieval performance in *RL*1 with the retrieval performances in *RL*2, *RL*3 and *RL*4, we can evaluate whether a retrieval system can use the history of user interaction prior to the current query to improve the retrieval performance for this query. Therefore only G1 is tested.

It should be noted that the names used for the ranked lists in Session Track 2010 and 2011 may represent different result lists and should not be mistaken. For example, *RL*1 of Session Track 2011 is effectively equivalent to *RL*2 of Session Track 2010 and *RL*2 in Session Track 2011 is effectively equivalent to *RL*3 in Session Track 2010.

7.1.3. Experimental Setup

In this section we describe the fundamental framework for conducting experiments on Session Tracks 2010 and 2011. The framework was used for our submissions as a group and for further experiments.

The Document Collection

The Session Track studies the session retrieval problems in the context of Web search engines and uses the ClueWeb09 as the target document collection. The ClueWeb09 dataset[1] is a Web crawl of more than a billion pages that has been used in the Web track of TREC 2009. The ClueWeb09 category B dataset is a subset of the larger ClueWeb09 crawl and it consists of the first 50 million English pages of the crawl. In the Session Track task participants were permitted to use either one of the two datasets. In our submissions and experiments we used the category B dataset.

Retrieving Documents from ClueWeb09

An existing Indri[2] index of the ClueWeb09 dataset is already available and searchable via a public web service[3]. The web service would enable us to issue queries in the Indri query language and retrieve the top documents returned by the search engine, thus removing the burden of indexing the data internally. The Indri search engine uses language modelling probabilities and supports query expansion.

We also used the Waterloo Spam Rankings[4] for the ClueWeb09 dataset to filter the spam documents from the returned ranked lists. We consider documents with scores of 70% or less as spam which is recommended by the creators of those rankings (Cormack et al., 2011).

The general procedure used to retrieve documents form the index is described in Algorithm 2. The procedure 'RetrieveFromIndriIndex' takes a query in the Indri language and a maximum number of results to retrieve documents via the Web service ranked according to the retrieval model implied in the Indri query. Note that we only consider the top 1000 documents retrieved

[1] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`
[2] `http://lemurproject.org/indri.php`
[3] `http://boston.lti.cs.cmu.edu:8085/clueweb09/search/cataenglish/lemur.cgi`
[4] `http://durum0.uwaterloo.ca/clueweb09spam/`

Algorithm 2: The Procedure of retrieving documents for a query.

Input: Indri formatted query q
Output: A ranked list of documents from ClueWeb09 RL

```
1 RL ← RetrieveFromIndriIndex(q,1000)
2 foreach d ∈ RL do
3 |   if ConfidenceScore(d) < 70 then RL ← RL \ {d}
4 end
```

from the index. The procedure 'ConfidenceScore' takes a document as an argument and returns its confidence score according to the Waterloo Spam Rankings mentioned above.

In the next sections describing our experiments, we will use the following terminology. For an Indri formatted query q, our reference retrieval model (The Indri retrieval model) would return a ranked list of documents using the previous procedure:
$D_q < d_{q,1}, d_{q,2}, ..., d_{q,n} >$ where $d_{q,i}$ refers to the document ranked i for the query q based on the retrieval model's ranking and after removing the spam documents.

Note that the Indri index uses the query maximum likelihood language model with Dirichlet smoothing to retrieve the top matching documents if no other operator is used (Strohman et al., 2004). For a query q containing a number of terms t_i, the documents are ranked according to their scores as defined in Equation 7.1.

$$Score(d,q) = \prod_{t_i \in q} P_{MLE}(t_i|d) \tag{7.1}$$

Where $P_{MLE}(t|d)$ is the maximum likelihood estimate of term t in document d under the given smoothing conditions as defined by the Indri model.

General Setup for Session Track 2010

In our submissions and experiments, we produce the first two ranked lists $(RL1)$ and $(RL2)$ by submitting the original query q consisting of a number of terms qt_i and the reformulation r consisting of a number of terms rt_i to the Indri index without any preprocessing, i.e. the first two ranked lists will be D_q and D_r respectively. The major challenge is producing the ranked list $RL3$ where both the current query and the previous query in the session can be used. We show how our models can be utilised for this task. Note that we do not aim to train our models with the session data provided by TREC (as test topics). As we have previously shown, the adaptive domain models need to be trained on large-scale query logs in order to be effective. Therefore, this data is not suitable as training data for our models. Instead, we aim to train our models on large-scale logs and use it to improve the effectiveness of session retrieval.

General Setup for Session Track 2011

In our submissions and experiments we generate $RL1$ by simply submitting a preprocessed version of the current query q'_m to the Indri index using the procedure in Algorithm 2, i.e. $RL1$ will be equivalent to $D_{q'_m}$.

The current query q_m was processed to produce q'_m following these steps:

1. Removing the following punctuation marks '(',')',',','?'.
2. Removing stop words from a common list of English stop words that do not fall within quoted text.
3. Replacing quotes with the corresponding Indri syntax #1(<quoted text>), e.g. *"event planning"* becomes *#1(event planning)*.
4. Replacing site specification with the corresponding Indri format, e.g. "female winemakers site:.com.au" becomes "female winemakers com.url au.url".

The reason for preprocessing the query as opposed to not performing any preprocessing for *RL*1 and *RL*2 in the Session Track 2010 is that the queries in 2011 were collected from real user sessions whereas in 2010 the queries were simulated.

Generating *RL*2 would be a similar task of generating *RL*3 in the Session Track 2010 task. However, the number of previous queries varies in the sessions. An additional challenge in the Session Track 2011 exists when producing the ranked lists *RL*3 and *RL*4 where previous interactions of the user can be utilised.

We show approaches for using our adaptive models to generate these lists. As in the previous task (previous year), we do not aim to train our models with the session data provided by TREC (as test topics), as it is not suitable for our models which need to be trained on large-scale query logs.

7.2. Anchor Log for Session Retrieval

Anchor text has shown to be effective for a variety of information retrieval tasks. This includes ad-hoc search and the diversity task (Craswell et al., 2009; Koolen and Kamps, 2010). Anchor text can be considered as a replacement to user queries as often web authors use similar labels to describe web pages to those used by searchers to find them (Eiron and McCurley, 2003). Moreover, Dang and Croft (2010) have recently shown how anchor text can be used to simulate user sessions. They have considered all the anchor text pointing to the same document as queries in the same user session. In this work, we adopted this technique to simulate query sessions. We propose to use the anchor logs as an alternative to large Web search logs, which are not easily accessible, to build query recommendation models similar to the ones we experimented with previously. These models are then employed for the session retrieval problem.

7.2.1. ClueWeb09 Anchor Logs

In our experiments, we aim to use anchor logs to simulate query logs. The mean reason for not using query logs and simulating with anchor logs instead is that query logs related to the tasks of the Session track were not provided by TREC. Moreover, large-scale query logs of commercial Web search engines may be hard to obtain. The anchor log for the dataset has been processed and made publicly available by the University of Twente (Hiemstra and Hauff, 2010). Each line in the log represents a document in the collection with all the anchor text pointing to the document. We used the anchor log file of the ClueWeb09 category B dataset. This file contains 43 million lines and thus contains anchor text for about 87% of the documents. Each line is tab separated and consists of the document TREC identifier, its URL and all the anchor text pointing to that document.

clueweb09-en0000-23-00060 *http://001yourtranslationservice.com/dtp/* 'website design' 'DTP and Web Design' 'Samples' 'programmers' 'desktop publishing' 'DTP pages' 'DTP samples' 'DTP and Web Design Samples' 'DTP and Web Design Samples' 'DTP and Web Design Samples' 'DTP and Webpage Samples' 'DTP' `http://001yourtranslationservice.com/dtp/`

Figure 7.1.: A sample of the anchor log file

Figure 7.1 shows a sample line in the anchor log file. We add quotation marks to group anchor text fields for illustration purposes.
We apply the same normalisation steps , as applied on queries in a search log as explained in Section 3.1.6, to all the anchor text. This includes case folding, punctuation removal, etc.

For a document d in the collection, we denote the unique set of strings that represent all the anchor text found in the collection pointing to this document is referred to as A_d. For a string $z \in A_d$, the number of times that z occurred as an anchor text for document d is denoted by $f_A(z,d)$

7.2.2. Recommendation Models using Anchor Logs

Our approach for the session retrieval problem is to perform query expansion that is related to the context of the entire session and not only an ad-hoc query. Here we develop a method for extracting useful terms and phrases to expand the *current query* in the session. As described in the previous sections we used an anchor log constructed from the same dataset (the ClueWeb09 category B dataset) to simulate query logs. We consider all the anchor text pointing to one document as a set of queries in a user session. Following these assumptions we can derive query recommendations for a user query using association rules proposed by Fonseca et al. (2003). Ideally we would like to employ our adaptive models presented in Chapters 3 and 4 to derive query recommendations. However this is not possible because the simulated user session consists of a set of queries (anchor text) not a chronologically ordered list of queries. Therefore we use the 'Association Rules' model as an alternative that can derive recommendations without knowing the order of queries in user sessions.

The intersection of recommendations extracted for queries in the *same* session can be considered useful for query expansion of the reformulated query as they can provide an approximation of the potential user session route.
Assuming that the user session is only represented with a pair of queries, the current query and a *previous query* in the session. The following steps were taken for each query pair to extract the query expansion terms and phrases:

- We remove stop words from an English common stop word list from both queries in the session.
- From the anchor log, we extract all the lines (the sessions) in which the anchor text contains either one of the queries.

- If one of the queries is a substring of the other one, i.e. the queries look like $q_a q_b$,q_a or vice versa, then we treat the pair as the pair (q_a, q_b). Where q_a, q_b represent two queries containing one term or more.

- Using the association rules approach for log data proposed by (Fonseca et al., 2003) we extract all the query recommendation for each of the queries q_a, q_b respectively.

  The sets of recommendations for both queries are denoted as $R(q_a), R(q_b)$. In this step, we considered some anchor terms as stop words and filtered out these stop words from the anchor text. This is due to the observation made previously by Eiron and McCurley (2003). that links within the site are often navigational links and they results in anchor terms such as 'click', 'next', 'here' (Eiron and McCurley, 2003).

- We intersect the two sets of recommendations in the previous step and consider those as candidate expansions to the *current query*. The candidate set of expansions is $E = R(q_a) \cap R(q_b)$.

- The final expansion set consists of the *previous query* and up to 10 queries from a ranked list of queries in the candidate expansion set E. The queries are ranked according to their score provided by the 'Association Rules' model for $R(q_b)$.

  In the Session Track 2010, to generate the ranked list *RL*3 for a the current query r in the session which contains a previous user query q, we submit the following query (using Indri belief operators to weight the different query terms):

$$\text{\# weight(}$$
$$0.7 \text{ \# combine(} rt_1\, rt_2 \,..\, rt_n)$$
$$0.3 \text{ \# combine(} q\, e_1\, e_2 \,..\, e_{10})$$
$$)$$

where rt_i are the individual terms in the reformulated query r, q is the query phrase of the original query q and $e_i \in E$ is an expansion term or phrase extracted as explained in the previous step. We empirically chose the values 0.7 and 0.3 for the expansions to assign more importance to the current query. Note that the Indri "#band" operator is used for phrases (where the number of terms is more than 1). Note that in the case where no expanded terms or phrases are extracted in the previous step, we are only expanding with the previous query.

Table 7.8 shows some the extracted expansion terms and phrases for 3 different pairs.

Session	**Expansion terms and phrases**
gps devices → garmin	'gps devices', 'wikipedia','usb', 'gps device', 'gps products', 'garmin nuvi880', 'garmin gps device','visit garmin'
computer worms → malware	'computer worms','computer security', 'category','worm'
us geographic map → us political map	'us political map','article'

Table 7.1.: Example of expansion terms and phrases extracted for three query pairs in the Session Track 2010

In the Session Track 2011, the same procedure is followed using a pair of queries in the session consisting of the first query q_1 and the current query in the session q_m. All intermediate queries are ignored.

7.2.3. Experiments in the Session Track 2010

Table 7.2 illustrates the ranked lists matrix of our three submissions as a participating group. In the following subsections we explain how we produced the ranked list *RL*3 for each run.

	RL1	RL2	RL3
essex1	D_q	D_r	(baseline 1)
essex2	D_q	D_r	(baseline 2)
essex3	D_q	D_r	(AnchorLog Approach)

Table 7.2.: The runs' matrix

The First Baseline - essex1

This baseline represents the simplest way of using previous user interaction with the search engine to interpret reformulated queries. This is done by submitting a new query $q+r$ to our search engine where the terms in this query is the set $qt \cup rt$. i.e. the system will return the ranked list D_{q+r} as $(RL3)$.

The Second Baseline - essex2

This baseline reflects on the assumption that the users are not satisfied with the first set of results and that is why they reformulated their original query. Therefore one possible naive way to utilise the previous query is to filter the results for the next query by eliminating whatever appears in the result set returned for the original (first) query. In this baseline, for the ranked list $(RL3)$ we return the ranked list: $D_r \setminus D_q = \{d; d \in D_r, d \notin D_q\}$
The documents in $D_r \setminus D_q$ are ordered using their ranking in D_r.

The Anchor Log Approach - essex3

This is the run we would like to test against the other two baselines. It is the approach where we extract expansion from the anchor log model as explained in Section 7.2.2.

Results and Discussion

Table 7.3 shows the overall retrieval performance of our 3 runs and the summary results of all the participants in the track. The normalised Discounted Cumulative Gain (nDCG) metric was estimated together with an extended version, the normalised session DCG (nsDCG), introduced in (Järvelin et al., 2008) to take into account multiple interactive queries. Like nDCG, nsDCG penalises documents that appear lower in the ranked list for a given query. However, it further penalises documents that appear later in the user session after query reformulations.

The figures in Column 2 represent the normalised session discounted cumulative gain $nsDCG(RL13)$ and they are used as the main measure for goal G2 (evaluating the performance over the entire session). Both $nsDCG(RL13)$ and $nsDCG(RL12)$ can be used to compare the performance of our runs for goal G1 (testing whether a system can improve its performance by using information from previous queries)

Run	*nsDCG.RL*12	*nsDCG.RL*13	*nDCG.RL*1	*nDCG.RL2*	*nDCG.RL3*
essex1	0.2154	0.2231	0.2077	0.2215	0.2348
essex2	0.2154	0.1993	0.2077	0.2215	0.1700
essex3	0.2154	**0.2246**	0.2077	0.2215	**0.2456**
min	0.0666	0.0458	0.0557	0.0900	0.0263
median	0.2044	0.1784	0.1894	0.2144	0.1700
max	0.2488	0.2375	0.2354	0.2658	0.2602

Table 7.3.: The results for our runs and the overall results of the Session Track. The figures in bold are the best scores in our runs for *nsDCG.RL*13 and *nDCG.RL3*.

We summarise the findings of analysing these results as follows:

- Our anchor expansion approach 'essex3' outperforms both baselines 'essex1' and 'essex2' for goal G2. Both 'essex3' and 'essex1' runs are among the top performing systems in the

	nsDCG@10			nDCG@10			
Run	RL12	→	RL13	RL1	RL2	→	RL3
essex3	0.2154	↑	0.2249	0.2077	0.2215	↑	0.2461
essex1	0.2154	↑	0.2234	0.2077	0.2215	↑	0.2353
essex2	0.2154	⇓	0.1993	0.2077	0.2215	⇓	0.1700

Table 7.4.: The ↑ symbol denotes a measurable but not significant increase in the performance of the retrieval system when utilising the initial query compared with the performance of the system when ignoring the initial query, while the ↓ denotes a drop. The ⇓ symbol denotes a significant drop when a paired two tailed t-test is applied

	nsDCG@10.RL13			
Run	all sessions	Specification	Generalisation	Drifting
essex3	0.2249	0.1481	0.2531	0.2763
essex1	0.2233	0.1456	0.2538	0.2738
essex2	0.1993	0.1395	0.2190	0.2416

Table 7.5.: Results showing performance over the entire session, RL1 → RL3 for the different reformulation types

track for goal G2 as their $nsDCG(RL13)$ score is close to the maximum score reported by NIST and is above the median.

- Each of 'essex1' and 'essex3' systems has achieved a marginal overall improvement of retrieval performance for $RL13$ over $RL12$ i.e. they were both capable of using previous queries to improve retrieval performance. The anchor expansion approach 'essex3' was marginally better than 'essex1'. However both approaches did not achieve a statistically significant improvement when a paired two tail t-test is applied on $nsDCG@10$ for $RL12$ and $RL13$. The second baseline 'essex2' failed to improve retrieval performance when using previous queries history. Table 7.4 illustrates the specifics for goal G1.

- To analyse the performance with respect to the reformulation type, Table 7.5 illustrates the results for goal G2 when considering each reformulation type. In all runs, the order of performance with regards to reformulation type is drifting, generalisation then specification. This suggests that drifting is the easiest reformulation type and specification

is the hardest. In particular the scores obtained for specification sessions were significantly lower from the ones obtained for generalisation and drifting.

We also analyse the performance according to the reformulation type for goal G1 in Table 7.6. For all runs, better results were achieved in drifting and generalisation and no improvement was obtained for specification sessions. Both 'essex3' and 'essex1' achieved improvement for *RL*13 over *RL*12 in drifting and generalisation sessions but not in specification. However when taking the average percentage increase into account in Table 7.7 the anchor expansion approach 'essex3' did improve on all reformulation types including specification.

- Expansion terms could not be derived for all query pairs using our anchor log approach. When looking at individual query pairs where 'essex3' succeeded in extracting query expansions from the anchor logs, the majority of these resulted in a better retrieval performance over 'essex1' for *RL*3. In 'essex3' we successfully obtained expansions for the reformulated query in 52 topics out of the 136 judged by NIST. In 69% of those topics 'essex3' achieved a better performance than 'essex1' with regards to the first task.

Run	nsDCG@10								
	Specification			Generalisation			Drifting		
	RL12	→	RL13	RL12	→	RL13	RL12	→	RL13
essex3	0.1563	↓	0.1481	0.2381	↑	0.2531	0.2542	↑	0.2763
essex1	0.1563	↓	0.1456	0.2381	↑	0.2538	0.2542	↑	0.2738
essex2	0.1563	⇓	0.1395	0.2381	↓	0.2190	0.2542	⇓	0.2416

Table 7.6.: System performance per reformulation type.

Run	All sessions	Specification	Generalisation	Drifting
essex3	19.83	0.20	14.37	44.36
essex1	7.32	-13.61	12.85	23.32
essex2	-6.67	-8.42	-1.67	-9.39

Table 7.7.: % average increase from nsDCG@10.RL12 to nsDCG@10.RL13

7.2.4. Experiments in the Session Track 2011

In our submission as a participating group in the Session Track 2011, we submitted 3 different runs. One of the runs 'essexAnchor' uses the Anchor Log expansion approach.

The 'essexAnchor' Run

In this run, to generate *RL*2 we use the same approach used in the 'essex3' run for the Session Track 2010. We consider the first query and the current query in the session q_1, q_m and use the same method described in Section 7.2.2 to generate query expansions from the anchor log to the current query q_m. Note that we discard all queries submitted between q_1 and q_m.

For *RL*3 and *RL*4, we extend the anchor log approach. In *RL*2 we model the user interests by finding potential related queries (anchor text) that best represent the user information needs as expressed in her queries throughout the session. However the interaction data in a user session also includes the documents that were displayed to the user and those which are clicked with their dwelling times. We aim to use this interaction data to create a better model of the user interests. We may consider the documents displayed throughout the session as good indicators of the user interests which is a similar approach to the pseudo relevance feedback model, e.g. (Rocchio, 1971). We may also consider only those clicked as useful indicators which is a similar approach to the implicit relevance feedback models, e.g. (Kelly and Belkin, 2001; Radlinski and Joachims, 2005).

The anchor log can then be used to derive expansions by considering anchor text that point to those documents as potential query expansions. The intuition is that they represent alternative queries relevant to the user's information needs as inferred from her interactions throughout the session. The fundamental difference of our method to the relevance feedback framework is that in effect we aggregate the pseudo or implicit relevance feedback throughout the session.

Therefore to generate *RL*3 and *RL*4, we generate a candidate expansion set as follows. Let D_{disp} be the set of displayed documents in the session. Let D_{clk} be the set of clicked documents in the session. For *RL*3, the candidate expansion set E_{RL3} is a union of all the anchor text of the displayed documents D_{disp} to the user, i.e. $E_{RL3} = \bigcup_{d \in D_{disp}} A_d$. For *RL*4, the candidate expansion set E_{RL4} is a union of all the anchor text of the clicked documents D_{clk} by the user $E_{RL4} = \bigcup_{d \in D_{clk}} A_d$.

In both cases, we expand the current query q_m with queries (anchor text) from the expansion set and use the Indri "#weight" belief operator to assign more weights for more dominant queries (anchor text) using their frequency across all the documents. We only used the top 10 queries (anchor text) for expansion.

The Indri query generated for both *RL*3, *RL*4 has the following syntax:

weight(

0.7 # combine(q_m)

0.3 # weight(w_1#e_1 w_2#e_2 .. w_{10}#e_{10})

)

where

- $e_i \in E_{RL3}, e_i \in E_{RL4}$ for *RL*3 and *RL*4 respectively;
- w_i is the frequency of the anchor text across the displayed documents (*RL*3, Equation 7.2) or clicked documents (*RL*4, Equation 7.3).

$$w_i = \sum_{d \in D_{disp}} f_A(e_i, d) \tag{7.2}$$

$$w_i = \sum_{d \in D_{clk}} f_A(e_i, d) \tag{7.3}$$

In Table 7.8 we give an example of the expansions extracted to generate queries for *RL*3, *RL*4 for session 63.

	Expansion:weight
RL3	united states foreign intelligence surveillance court:0.31 foreign intelligence surveillance act:0.23 united states foreign intelligence surveillance court of review:0.10 fisc:0.08 fisa:0.07 section summary of the usa patriot act title ii:0.06 usa patriot act title ii:0.05 ii:0.03 title ii enhanced surveillance procedures:0.03 enhanced surveillance procedures:0.03
RL4	foreign intelligence surveillance act:0.78 fisa:0.16 article:0.02 foreign intelligence surveillance act fisa:0.02 http en wikipedia org wiki foreign intelligence surveillance act:0.02

Table 7.8.: Example of expansion terms and phrases extracted for session 63: FISA → judges on fisa court → 1990 FISA wiretap applications → judges FISA court → judges FISA court 2005

Results and Discussion

Tables 7.9 and 7.10 summarise the results for our run as well as the maximum and median for all the systems submitted to the Session Track 2011.

Unlike the Session Track 2010, NIST assessors provided relevance assessments on two different criteria thus the results in Table 7.9 take into account all the subtopics, whereas the results in Table 7.10 only reflect the user's last subtopics.

Each of the table columns represents the normalised discounted cumulative gain (*nDCG*) for each result list submitted. Despite receiving a variety of relevance metrics we will focus our results analysis on the *nDCG* results, as they relate to the established metrics from last year's competition. The tables are split in two, the top half using the *nDCG* metric for all the documents submitted and the bottom showing the *nDCG* score using only the first ten returned documents (*nDCG*@10).

The arrows in the *RL*2, *RL*3 and *RL*4 columns represent the relative improvement or decline in the nDCG scores between the results lists for a given system (row), with a double (⇑) arrow indicating that a paired two tailed t-test has supported the result as significant. For instance an upward arrow (↑) in the *RL*2 column indicates that *RL*2 improves on RL1, and the first and second arrows in the *RL*3 column compare *RL*3 to *RL*1 and *RL*3 to *RL*2 respectively. These results are also charted without significance test data in the graphs in Figure 7.2 together with our other two runs which will be discussed in the next section. The following subsections outline the results obtained for each system. In cases where we do not explicitly refer to a result as significant it can be assumed that the comparison has returned a paired two tailed t-test value $p > 0.05$.

Taking all the subtopics (Table 7.9) we see that the retrieval performance of *RL*2 is worse than *RL*1 when looking at all retrieved documents, but conversely the *nDCG*@10 score does improve i.e. the top results improve. This is in line with the results of Session Track 2010 where expansions using association rules from query logs simulated from anchor logs improved performance. However, using only the last subtopic *RL*2 performs worse than *RL*1 on both *nDCG* and *nDCG*@10.

System	RL1.nDCG	RL2.nDCG	RL3.nDCG	RL4.nDCG
max	0.3433	0.3353	0.3993	0.4112
median	0.2804	0.2918	0.2363	0.2577
essexAnchor	0.2669	$\downarrow$ 0.2595	$\Uparrow\Uparrow$0.2923	$\Uparrow\Uparrow\downarrow$ 0.2866
System	RL1.nDCG@10	RL2.nDCG@10	RL3.nDCG@10	RL4.nDCG@10
max	0.3789	0.4281	0.4307	0.4540
median	0.3232	0.3215	0.3259	0.3407
essexAnchor	0.3634	$\uparrow$ 0.4016	$\Uparrow\uparrow$ **0.4307**	$\Uparrow\uparrow\downarrow$ 0.4175

Table 7.9.: *nDCG* values when assessing over all subtopics; the arrows indicate improvement($\uparrow$) or decline ($\downarrow$) against the previous results lists, the first arrow in a cell relates to *RL*1, the second arrow to *RL*2 and so on. Double arrows ($\Uparrow$ / $\Downarrow$) indicates the comparison is statistically significant returning a paired two tailed t-test $value \leqslant 0.05$. The figure in bold is the top obtained score for that measure in Session Track 2011.

Comparing *RL*3 to *RL*1 demonstrates significant improvement in all cases, i.e. when using the last subtopic, all the subtopics and against *nDCG* and *nDCG*@10. Therefore the implicit relevance feedback from anchor text of retrieved documents does improve the retrieval performance. In fact our anchor log approach was the top performing system among all systems submitted in the Session Track 2011.

*RL*4 also shows improvement relative to *RL*1 in all cases, however it is only significant when using all subtopics. *RL*3 is marginally better than *RL*4 in all cases though in general the difference is so small as to consider them on par. Thus using the anchor text only from clicked documents to expand queries does not appear to improve performance.

Relevance scores for our non-baseline submissions (*RL*3 and *RL*4) are between median and maximum of the composite scores for the track and when using *nDCG*@10, their performance is very close to maximum and well above median results reported. In fact 'essexAnchor' was the top performing system among all the participants when it comes to *nDCG*@10 for *RL*3 using all subtopics. The anchor text approach performs better when assessed over all the subtopics than when measured against only the last subtopics, which is not surprising as this method tries to model user interest throughout the session and does not only target a specific subtopic of a query. Note that this correlates with the overall maximum and median scores.

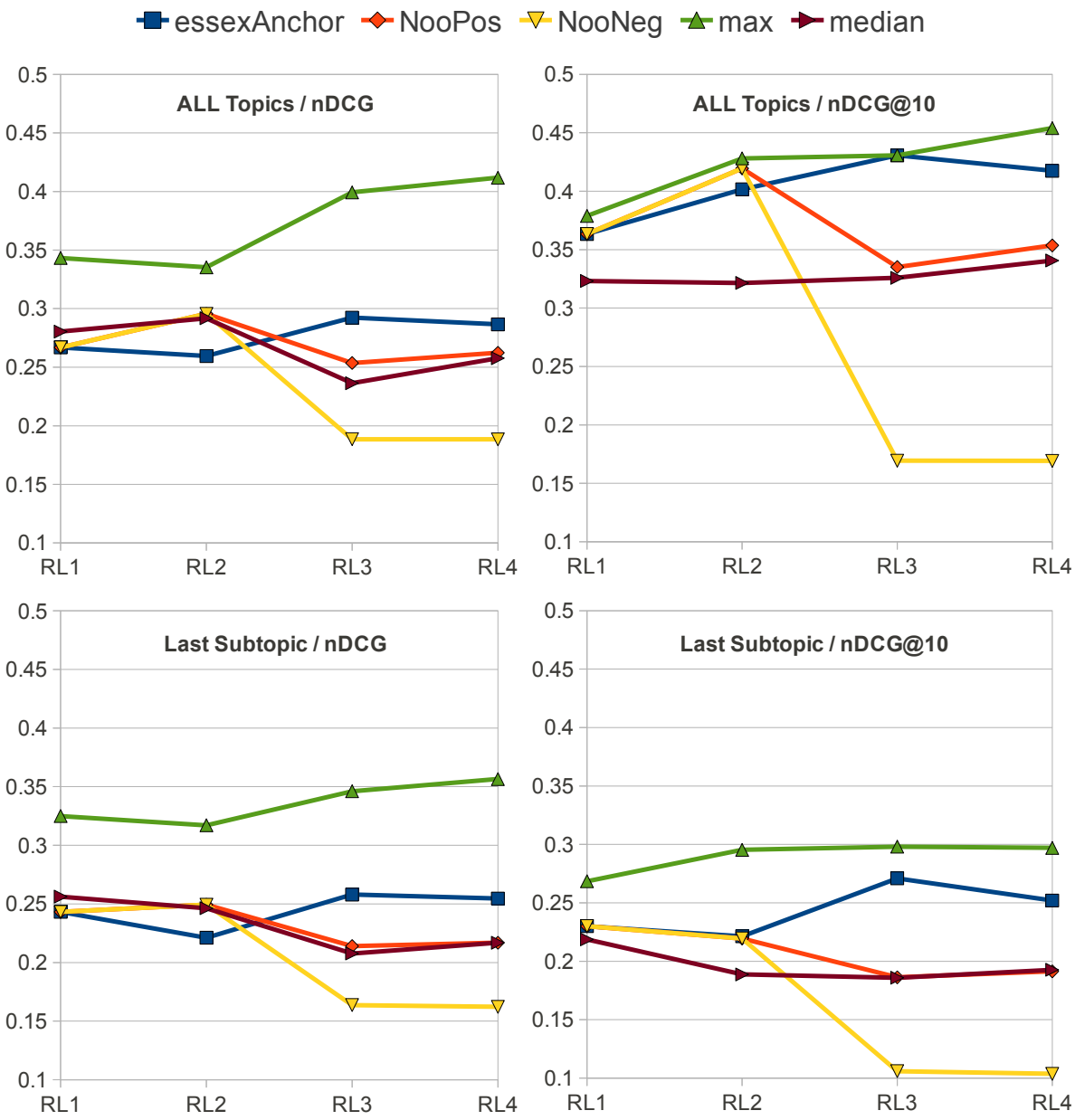


Figure 7.2.: Graphs showing nDCG and nDCG@10 results of the Essex group runs

System	RL1.nDCG	RL2.nDCG	RL3.nDCG	RL4.nDCG
max	0.3249	0.3170	0.3461	0.3565
median	0.2562	0.2463	0.2077	0.2169
essexAnchor	0.2432	$\downarrow$ 0.2211	$\Uparrow\Uparrow$ 0.2580	$\uparrow\Uparrow\downarrow$ 0.2546
System	RL1.nDCG@10	RL2.nDCG@10	RL3.nDCG@10	RL4.nDCG@10
max	0.2685	0.2954	0.2981	0.2971
median	0.2187	0.1888	0.1859	0.1927
essexAnchor	0.2301	$\downarrow$ 0.2214	$\Uparrow\Uparrow$ 0.2710	$\uparrow\uparrow\downarrow$ 0.2520

Table 7.10.: nDCG values when assessing the last subtopic(s); the arrows indicate improvement($\uparrow$) or decline ($\downarrow$) against the previous results lists, the first arrow in a cell relates to RL1, the second arrow to RL2 and so on. Double arrows ($\Uparrow$ / $\Downarrow$) indicates the comparison is statistically significant returning a two tailed t-test $value \leqslant 0.05$.

7.3. Modelling Search Sessions with Nootropia

In Chapter 3 we have shown how we built log-based domain models for query recommendation using Nootropia, where the network we build aggregate the interests of the user population from their search history over time. Here we use Nootropia differently as the idea is to build and update a profile for an individual user from her previous interactions within the same session.

As discussed previously, Nootropia is a biologically inspired Information Filtering system that has been used successfully in news recommendation (Nanas et al., 2009). With Nootropia, information needs are represented as a weighted and ordered network that may represent an individual's multiple topics of interest. This network profile presentation can evaluate the relevance of an information item to the user interests based on a directed spreading activation model. Nodes (features) in the network that appear in the item are activated and subsequently disseminate part of their initial energy to nodes with larger weight. At the end of this feedforward dissemination process, a single relevance score can be calculated as the weighted sum of the final energies of activated terms.

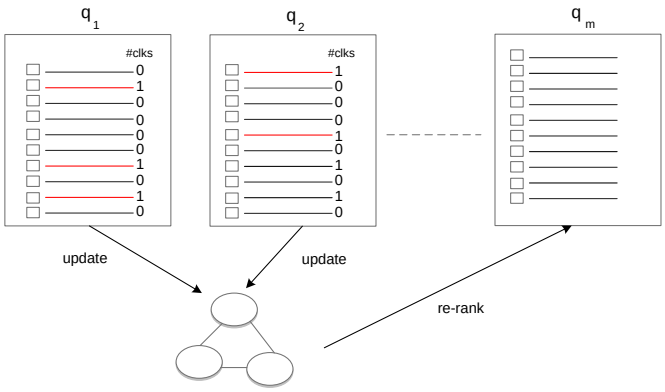


Figure 7.3.: Illustration of the process of applying Nootropia

7.3.1. Nootropia to Model Query Sessions

We study Nootropia in the context of the Session Track 2011. The aim is to use Nootropia to adapt a profile of information needs throughout the session and exploit the profile in the retrieval process for the current query in the session.

The proposed process of using Nootropia to generate *RL*3 and *RL*4 is illustrated in Figure 7.3. Throughout the session we take the documents that have been displayed to the user by the search engine to create a Nootropia model. For the current query q_m, we submit the preprocessed version q'_m, as explained before, to the search engine and then we use the profile built so far in a form of a Nootropia network to give a score to the returned documents reflecting how much they match the profile. The scores can be used to re-rank the documents. We used two routes to re-rank the documents using the Nootropia profile.

In our submission as a participating group in the Session Track 2011 and in addition to the aforementioned 'essexAnchor' run, we proposed two other runs that cover both routes for re-ranking documents using the built Nootropia profile.

- **Nootropia Positive (essexNooPos)**: To generate *RL*3, we can use all the displayed documents each time to build a Nootropia profile and update it. Here we take an "optimistic" approach and consider displayed documents as useful documents to the user. The doc-

uments that match this profile should be promoted. Therefore we rank the documents returned by the search engine for the current query by their Nootropia score in an *descending* order and that would be our *RL*3.

Let N^a be the Nootropia profile for all displayed documents to the user.
Let $S_{N^a}(d)$ be the matching score of a document d in the profile N^a.

$$RL3 = \langle d_1, d_2, .., d_n \rangle; \forall i : d_i \in D_{q'_m} \wedge S_{N^a}(d_i) \geq S_{N^a}(d_{i+1})$$

For *RL*4, we take only the clicked documents to build a profile of user interests unlike the previous profile.

Let N^+ be the Nootropia profile for all clicked documents by the user.
Let $S_{N^+}(d)$ be the matching score of a document d in the profile N^+.

$$RL4 = \langle d_1, d_2, .., d_n \rangle; \forall i : d_i \in D_{q'_m} \wedge S_{N^+}(d_i) \geq S_{N^+}(d_{i+1})$$

- **Nootropia Negative (essexNooNeg)**: Here we take a "pessimistic" approach and consider the displayed documents not useful to the user as they had to reformulate the query and therefore any document that matches this profile should be penalised. Therefore we rank the documents returned by the search engine for the current query by their Nootropia score in an *ascending* order and that would be our *RL*3.

  Let N^a be the Nootropia profile for all displayed documents to the user.
  Let $S_{N^a}(d)$ be the matching score of a document d in the profile N^a.

  $$RL3 = \langle d_1, d_2, .., d_n \rangle; \forall i : d_i \in D_{q'_m} \wedge S_{N^a}(d_i) \leq S_{N^a}(d_{i+1})$$

  As for *RL*4, we take only the documents not clicked on to build a profile of documents that the user is not interested in. The documents that match this profile should be penalised. Therefore, we rank the documents returned by the search engine for the current query by their Nootropia score in an *ascending* order and that would be our *RL*4.

Let N^- be the Nootropia profile for all abandoned (i.e. not clicked on) documents by the user.

Let $S_{N^-}(d)$ be the matching score of a document d in the profile N^-.

$RL4 = \langle d_1, d_2, .., d_n \rangle; \forall i : d_i \in D_{q'_m} \wedge S_{N^-}(d_i) \leq S_{N^-}(d_{i+1})$

To update the profile, we should feed the document or a representation of the document to the Nootropia model. For this we extract all the nouns and noun phrases in the snippet of the document. For the detection of noun phrases we look for particular patterns of sequence of part-of-speech tags based on the algorithm for the detection of terminological terms as described in (Justeson and Katz, 1995).

In both runs, to generate *RL*2 we use a similar approach to the one used in our baseline system in 2010, i.e. 'essex1' (see Section 7.2.3), where we used the previous history of the user by simply submitting both queries in the session. Therefore, to generate *RL*2 for both Nootropia runs we simply submit a query which consist of the first query and the last query in the session q_1, q_m.

7.3.2. Results and Discussion

Tables 7.11 and 7.12 summarise the results for both Nootropia runs.

When assessing all subtopics *RL*2 is significantly better than *RL*1 for both *nDCG* and *nDCG*@10. *RL*2 combines the user's first and last query and it was used as our baseline in Session Track 2010. The results here are in line with Session Track 2010 where this baseline has shown to be capable of improving the retrieval performance over an ad-hoc system. However considering only the last subtopic, *RL*2 improves on *RL*1 for *nDCG* but is worse for *nDCG*@10.

Comparing the relevance scores for *RL*3 and *RL*4 to *RL*1 shows performance declines when using Nootropia Positive using either evaluation strategy and in some cases the difference is

significant. While *RL*3 and *RL*4 both show a similar downward trend in comparison with *RL*1, only considering the clicked documents in *RL*4 does improve on *RL*3. Like our anchor test method the retrieval performance using all sub-topics is better than just using the last subtopics. This follows the general trend seen in the summary results provided by NIST, i.e. the maximum and median scores illustrated in tables 7.9 and 7.10.

The results for the Nootropia Negative run (*RL*3 and *RL*4), show that the "pessimistic" approach performs significantly worse than our baselines (*RL*1 and *RL*2) across all metrics. In addition using the click data (*RL*4) results in a further decline in retrieval performance. Comparing the results for Nootropia Negative (both *RL*3 and *RL*4) to Nootropia Positive we see that Nootropia Positive performs significantly better.

System	RL1.nDCG	RL2.nDCG	RL3.nDCG	RL4.nDCG
essexNooPos	0.2669	⇑ 0.2956	↓ ⇓ 0.2536	↓ ⇓ ↑ 0.2623
essexNooNeg	0.2669	⇑ 0.2956	⇓ ⇓ 0.1885	⇓ ⇓ ↓ 0.1884
System	RL1.nDCG@10	RL2.nDCG@10	RL3.nDCG@10	RL4.nDCG@10
essexNooPos	0.3634	⇑ 0.4195	↓ ⇓ 0.3351	↓ ⇓ ↓ 0.3536
essexNooNeg	0.3634	⇑ 0.4195	⇓ ⇓ 0.1694	⇓ ⇓ ↓ 0.1692

Table 7.11.: *nDCG* values when assessing over all subtopics; the arrows indicate improvement(↑) or decline (↓) against the previous results lists, the first arrow in a cell relates to *RL*1, the second arrow to *RL*2 and so on. Double arrows (⇑ / ⇓) indicates the comparison is statistically significant returning a two tailed t-test $value \leqslant 0.05$.

System	RL1.nDCG	RL2.nDCG	RL3.nDCG	RL4.nDCG
essexNooPos	0.2432	↑ 0.2493	⇓ ⇓ 0.2140	⇓ ⇓ ↑ 0.2169
essexNooNeg	0.2432	↑ 0.2493	⇓ ⇓ 0.1637	⇓ ⇓ ↓ 0.1622
System	RL1.nDCG@10	RL2.nDCG@10	RL3.nDCG@10	RL4.nDCG@10
essexNooPos	0.2301	↓ 0.2195	↓ ↓ 0.1863	↓ ↓ ↑ 0.1914
essexNooNeg	0.2301	↓ 0.2195	⇓ ⇓ 0.1059	⇓ ⇓ ↓ 0.1037

Table 7.12.: nDCG values when assessing the last subtopic(s); the arrows indicate improvement(↑) or decline (↓) against the previous results lists, the first arrow in a cell relates to RL1, the second arrow to RL2 and so on. Double arrows (⇑ / ⇓) indicates the comparison is statistically significant returning a two tailed t-test $value \leqslant 0.05$.

Overall both strategies failed to improve over the ad-hoc baseline where no previous interactions of the user is considered. The reason could be the small size of the pool of documents considered for scoring with the built profile. Ideally all documents in the collection should be matched against the profile but since it is not computationally feasible we only consider the baseline ranked list for re-ranking. Moreover, since Nootropia is originally designed for information filtering where richer explicit user feedback is provided to the system over a period of time. In our case, the sparse and limited feedback provided throughout the session may not be sufficient to build a representative profile.

7.4. Search Shortcuts for Session Retrieval

Derived from large query logs, Search Shortcuts have proven to be an effective model for query recommendation, especially for rare or unseen queries (Broccolo et al., 2012). In this section, we propose to use yet another log-based model for query recommendation to tackle the session retrieval problem. We show how the Search Shortcut model can be utilised to derive session-related query expansions. First we introduce the model and then we explain how we applied it for the Session Track 2011 task and study in detail the different factors that influence the session retrieval performance. The work introduced in this section is an outcome of collaboration with the ISTI-CNR group who provided their Search Shortcut platform to perform the experiments.

7.4.1. The Search Shortcut Recommender System

The Search Shortcut recommender system is a log-based query recommendation system that makes use of *successful* sessions (i.e. sessions ending with at least a click on the last query) present in a query log to produce query recommendations.

Let $\mathscr{L}$ be a search log of a Web search engine as explained in Section 3.1.2. We say that a session S is *successful* if and only if the user has clicked on at least one link shown in the result page returned by the search engine for the final query q_n and *unsuccessful* otherwise. Studies show that success of search goals correlates well with clicks on actual search results e.g. (Hassan et al., 2010).

The Search Shortcut algorithm (Broccolo et al., 2012) works by efficiently computing similarities between partial user sessions (the one currently being performed) and historical successful sessions recorded in a query log. Final queries of most similar successful sessions are suggested to users as search shortcuts. The algorithm is thus able to recommend queries by taking into account the activity of the user performed in the entire session (history-based recommendation). In particular, the algorithm works by computing the value of a scoring function δ, which for each successful session measures the similarity between its queries and a set of terms τ representing the user need expressed so far. Intuitively, this similarity measures how much a previously seen session overlaps with the current user need (the concatenation of terms τ serves as a bag-of-words model of the user need). Sessions are ranked according to δ scores and from the subset of the top ranked sessions, it is possible to suggest their final queries. It is obvious that depending on how the function δ is chosen the algorithm provides different recommendations. In (Broccolo et al., 2012), authors opted for δ to be the similarity computed as in the BM25 metrics (Robertson and Zaragoza, 2009).

In particular, authors define the final recommendation scoring formula as a linear combination of the BM25 score and the frequency in the log of the recommendation they are suggesting. Given a query q and the set of possible recommendations $R(q)$ for q, for each $r \in R(q)$, let V_r be the representation of the successful sessions ending with r. Therefore, the score of r is computed as follows:

$$\delta(q,r) = \alpha \times BM25(q,V_r) + (1-\alpha) \times f(r) \tag{7.4}$$

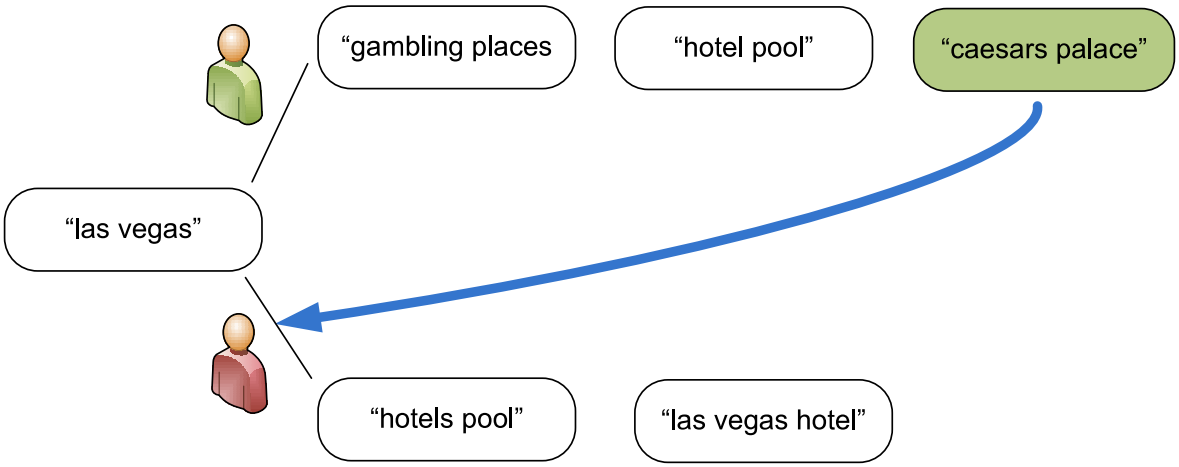


Figure 7.4.: Illustration of the Search Shortcuts method.

Figure 7.4 illustrates how the Search Shortcuts are utilized for query recommendation. Here, a previous user whose final query was “caesars palace” had a successful session; that is, the user clicked on at least one of the results returned by the search engine for the final query. We can therefore reduce the duration of a search session for a new user by suggesting the final query from the previous successful session with identical queries. As shown in the figure, rather than reformulating the initial query to “hotels pool” and “las vegas hotel”, “caesars palace” is more likely to direct the user to a landing page with more relevant results thereby (hopefully) creating another successful session.

The Search Shortcut was firstly proposed in (Baraglia et al., 2009) and then effectively designed in (Broccolo et al., 2012). The technique is not only generally efficient and very effective but also works well for queries in the long tail of the distribution (Broccolo et al., 2012). Here we are presenting another useful application of this model.

7.4.2. Deriving Session-related Expansions with Search Shortcuts

Unlike our log-based query recommendation models studied so far, the Search Shortcut technique is able to use “session” information to devise recommendations for a given query. This is

practically due to the organization of the knowledge model which makes use of an inverted index for computing recommendations.

Therefore we can use the Search Shortcut recommender system to derive relative recommendations for the current query in a given session. Furthermore, we propose that these query recommendations can be useful query expansions for the current query in the session.

We apply this technique in the context of the Session Track 2011 task and study different factors that affect its retrieval performance.

Search Shortcuts in Session Track 2011

Here we illustrate our proposals to apply Search Shortcuts as a query expansion technique for the task of the Session Track 2011.

We use the Microsoft RFP 2006 query log[5] to derive query recommendation with Search Shortcuts. The query log was preliminarily preprocessed by converting all queries to lower-case, and removing stop-words and punctuation/control characters. The queries in the log were then sorted by user and time-stamp, and segmented into sessions on the basis of a splitting algorithm which simply groups in the same session all the queries issued by the same users in a time span of 30 minutes (Radlinski and Joachims, 2005). Noisy sessions, likely performed by software robots, were removed. This is done by eliminating sessions with a large number of queries. The remaining entries correspond to approximately nine million (9M) sessions.

We build *RL*2 by using an expansion of the current query made by the terms composing the first three recommendations provided by the method and by uniformly weighting them after performing a stopword removal step.

Furthermore, *RL*3 has been built starting with an *arbitrary* number of recommendations generated for each current query of the sessions provided. We use the terms composing them to

[5] `http://research.microsoft.com/en-us/um/people/nickcr/wscd09/`

```
#weight(
0.7 #combine(event planning college)
0.3 #weight(0.16 #combine(college) 0.01
#combine(fashion) 0.36 #combine(event)
0.4 #combine(planning) 0.01
#combine(conference) 0.05 #combine(online) 0.01
#combine(management))
)
```

Figure 7.5.: An example of an expanded query with Search Shortcuts.

produce an expanded query representation. In particular, after removing stopwords, we develop a term weighting scheme based on how many times any given expansion term appears in the snippets of the documents returned within the session. The final weight is thus computed by dividing the frequency of the single expansion term with the sum of the frequencies of the expansion terms over all the returned documents.

More formally, let E be the set of the expansion terms obtained by the recommendations produced by Search Shortcuts. Furthermore, let D be the set of documents returned for all the queries of the current user session. Let $f_D(t)$ be a function measuring the frequency of the term $t \in E$ in the set of the snippets returned for the documents in D. The expansion weight of the term t used within *RL3* is thus derived using Equation 7.5.

$$w_t^{RL3} = \frac{f_D(t)}{\sum_{\forall x \in E} f_D(x)} \tag{7.5}$$

To illustrate the expansion process, Figure 7.5 shows a generated Indri query for session 2 (*RL3*). Note that for all the sessions and each ranked list, in line with what was suggested earlier, we chose the values 0.7 and 0.3 for both components of the expansion, i.e., the original query and the weighted expansion set of terms produced by the Search Shortcuts method.

Expansion term	RL3 weight	RL4 weight
court	0.7	0.85
judges	0.12	0.06
district	0.05	0.02
judge	0.02	0.05
united	0.01	0.01

Figure 7.6.: Sample expansion terms extracted for session 63: {FISA → judges on fisa court → 1990 FISA wiretap applications → judges FISA court → judges FISA court 2005}

*RL*4 has been produced starting from the weights obtained for *RL*3 and by adding to the terms appearing in the snippets of the clicked documents a boosting factor. This boosting factor depends on the frequency (with repetitions) of the given term in the set of the clicked documents divided by the total frequency of the expansion terms that are present in the set of the clicked documents. More formally, let C be the set of the clicked documents for all the queries of the current user session. Clearly, $C \subseteq D$. w_t^{RL4} can be thus computed by applying Equation 7.6.

$$w_t^{RL4} = w_t^{RL3} + \left(\frac{f_C(t)}{\sum_{\forall x \in E} f_C(x)} \right) \tag{7.6}$$

Figure 7.6 compares the expansion weights for session 63 across *RL*3 and *RL*4. It can be observed that the boosting factor increases the weights of some expansion terms while reducing the weights of others with respect to their RL3 weights after normalisation. Thus, in this example the RL4 weights of 'court' and 'judge' are boosted due to their frequency in the set of clicked documents.

It should be noted that for both *RL*3 and *RL*4, the number of recommendations we start with affects the retrieval performances. Our experiments show that best results are obtained when using 10 recommendations when compared with 3, 5, or 20 recommendations to start with.

7.4.3. Experiments in the Session Track 2011

Following the previous procedure for applying Search Shortcuts to generate the different ranked lists for each session in task of the Session Track 2011, we conducted a number experiments to answer the following questions:

1. Are session-related query expansions derived from large query logs of a Web search engine useful for the session retrieval problem?
2. How do the different ranking schemes of the Search Shortcuts and the session retrieval performances correlate? Is it the retrieval score or the frequency of the query in the log that matters?
3. How do different 'classes' of recommendation ('history-based' recommendation, 'current-query-based' recommendation) affect the session track performances?

Experiments on the Ranking Schemes

The ranking of the Search Shortcuts is given in Equation 7.4. The parameter α balances the importance given to either the frequency of the Search Shortcut or the retrieval score of similar sessions to user queries. We want to test the effect of the ranking schemes on the session retrieval performance. We test three values of α (1.0, 0.5, 0.0) which correspond to only BM25 ranking, equal weights on BM25 and frequency and only frequency ranking. We report the results of evaluation using all the subtopics criteria in table 7.13. Performances are evaluated in terms of $nDCG@10$ against all subtopics.

In all cases, we see that expansion with Search Shortcuts recommendations improve the retrieval performance significantly over the baseline $RL1$. In line with our anchor log expansion model, we also see that displayed and clicked documents do add value to the expansion process as the retrieval performance goes up.

Ranking Search Shortcuts for query expansions works best with BM25 scoring only while using only frequency harms the performance. Sessions which are closer (more relevant) to the user query are good candidates for query expansions to provide a better session retrieval performance. BM25 scoring for RL2 outperforms the other two schemes significantly. The increase in retrieval performance was not as significant over *RL*3 or *RL*4 which is expected as in *RL*3 and *RL*4 we use the same documents to filter the candidate expanding terms. Here we do not report the results for last subtopic evaluation, but we also found out that the BM25 scoring was the only model showing improvement.

System	RL1	RL2	RL3	RL4
BM25(α=1.0)	0.3634	$\Uparrow$ 0.3978	$\uparrow\uparrow$ 0.3981	$\Uparrow\uparrow\uparrow$ 0.4035
BM25+f(α=0.5)	0.3634	$\uparrow$ 0.3707	$\uparrow\uparrow$ 0.3965	$\Uparrow\Uparrow\uparrow$ 0.3993
f(α=0.0)	0.3634	$\downarrow$ 0.3579	$\uparrow\Uparrow$ 0.3854	$\Uparrow\Uparrow\Uparrow$ 0.3971

Table 7.13.: nDCG@10 values when assessing all subtopics; the arrows indicate improvement($\uparrow$) or decline ($\downarrow$) against the previous results lists, the first arrow in a cell relates to RL1, the second arrow to RL2 and so on. Double arrows ($\Uparrow$ / $\Downarrow$) indicates the comparison is statistically significant returning a two tailed t-test $value \leqslant 0.05$.

Search Shortcuts with 'No History'

In the previous experiment, the Search Shortcuts used for expansions are extracted using the entire session rather than the current query. We want to see the impact of using only the last query to extract shortcuts on the session retrieval performance. The intuition here is that recommendation with 'no history' may possibly improve the retrieval performance for the subtopics of the current query. However the results in Table 7.14 do not support that claim. One explanation could be that it is adding more noise to the current user interest as it diversifies the information needs. This behavior of the Search Shortcuts recommender system has been, in fact, already exploited within another application: Web search results diversification (Capannini et al., 2011). Here, the 'no history' recommendation has been used for generating a set of possible meanings behind a given 'ambiguous' query.

System	RL1	RL2	RL3	RL4
History	0.2301	↑ 0.2412	↑ ↓ 0.2332	↑ ↓ ↑ 0.2369
No History	0.2301	↓ 0.203	⇓ ⇓ 0.1583	⇓ ⇓ ↑ 0.1633

Table 7.14.: nDCG@10 values when assessing last subtopics; the arrows have similar indication of arrows in Table 7.13.

7.5. Concluding Remarks

In this chapter, we demonstrated the power of applying the methods for adaptive domain modelling, introduced in Chapters 3 and 4, in solving a practical task. More specifically, we discussed how a number of log-based domain models can be utilised to allow a retrieval system to better understand the user information needs throughout the session. The Session Track 2010 and 2011 allowed us to do extensive experimentation with three different models. Here we first give an overview of other techniques used by participants in both tracks. Then we give a summary of our findings.

7.5.1. Summary of the Session Track 2010 and 2011

To give the reader the state-of-the-art for the session retrieval topic, we briefly review the approaches used by the participating research groups in the Session Track. The Session Track Overview documents describe the methodologies participants employed in their submitted runs (Kanoulas et al., 2011, 2012). An analysis of these summaries and the related results achieved indicate the 'state-of-the-art' in search over session queries. In the Session Track 2010, methods used in creating ranked list RL3 included term weighting, pseudo-relevance feedback, ranked list re-ranking and merging and query expansion. The latter features prominently in the more successful submissions, including that from our group. Resources which participants used to help in the task of deriving expansion terms for ranked list RL3 included: query logs, anchor logs, Open Directory Project (ODP) categories and WordNet. The second year has seen a

near-complete overhaul of the track in terms of topic design, session data, and experimental evaluation. In the 2011 edition, the methods presented include pseudo-relevance feedback, query expansion, adaptive modelling, and semantic approaches. As in 2010, the resources used by participants to help in the task of deriving expansion terms consist of query logs, anchor logs and also external sources of information, like Wikipedia.

7.5.2. Our Findings

The Session Track provided a platform to evaluate the effectiveness of IR systems in interpreting query reformulations and previous user interactions throughout the session.

The adaptive models introduced earlier in the thesis were revisited and applied in the session context.

Simulating query logs, the anchor logs were used to derive query expansions for the current query in the session using the 'Association Rules' model. In the Session Track 2010, this achieved a better performance over a baseline system. Our anchor expansion approach was among the top performing systems and it has the best retrieval performance for both goals G1 and G2 when compared to the two baselines submitted. This finding was supported by the results of the Session Track 2011, and in addition to that, with interaction data the retrieval performance is further enhanced by using relevance feedback from the anchor text of retrieved documents. This approach achieved the best retrieval performance for RL3 among all participating systems in the track.

We also show how Nootropia, an adaptive IF system, can be employed for the session retrieval problem using two different strategies. For the proposed setup, no improvement was observed using either strategies. This is possibly due to the small size of the pool of documents matched against the learnt profile. This leaves a room for future work to investigate better techniques to use the learnt profile in re-ranking or retrieving relevant documents.

Finally we also experimented with another log-based query recommendation model namely Search Shortcuts. Mined from large Web query logs, the Search Shortcuts model was capable of generating session-related query expansions to guide session retrieval. The expansion with Search Shortcuts outperformed the ad-hoc retrieval baseline. We also show that similar sessions are more useful than popular queries for the expansion process. Moreover, we found that shortcuts which do not take into account previous actions of users has a negative impact on session retrieval.

Chapter 8.

Conclusions and Future Directions

In this Chapter, we finish the thesis by providing a summary of the materials introduced, highlight our contribution and discuss the opportunities for future work.

8.1. Summary

In this section, we summarise how the thesis addressed the research questions identified in Section 1.2

Can query logs be used as as a source of implicit feedback to build and adapt domain models that reflect the search behaviour of the user population in a given domain at a certain point of time?

The implicit user feedback captured in search logs is a rich source of knowledge that can be fed back into retrieval systems in order to improve the quality of the services they offer. There is a wide spectrum of methods and applications of utilising the implicit feedback for information retrieval. The theme of this thesis can be summarised as employing implicit user feedback captured in search logs for adaptive domain modelling. These adaptive domain models are then

employed to aid users of retrieval systems, in particular search engines in specific domains such as enterprises or digital libraries.

Domain modelling is a broad concept of structuring knowledge embedded in information materials within a specific domain of interest. It has attracted a lot of interest in various disciplines for different applications (see Chapter 2). The *Domain modelling* we studied in this thesis is the process of consuming the logs of all past user interactions recorded by the search engine to build data structures that mainly identify the queries in the domain and relations between the queries. These relations have no semantics and they only embed information about how related the queries are. They may also identify the more successful routes to good search results for similar information needs. They are also dynamic and they change over time to reflect the current search trends of the population of all users.

The implicit user feedback employed for adaptive domain modelling is rich as it consists of a variety of signals. In the thesis we presented a number approaches for adaptive domain modelling which use different types of these signals. First, we presented adaptive modelling techniques which *only* use the sequence (flow) of *queries* identified in individual user sessions. Two biologically inspired models were introduced to build adaptive domain models from query flows in Chapter 3. The ant colony optimisation algorithm was applied to build an adaptive domain model as a directed weighted graph of queries where edge weights are updated over time in a periodic offline process. The second model applies an immune inspired information filtering system (Nootropia) where a weighted network of queries is constructed and adjusted over time in a similar fashion to self organised antibody networks of the immune system.

Can clickthrough data be interpreted such that it is used as an implicit feedback to enhance the quality of the adaptive domain models?

Additional signals from implicit user feedback were further investigated to build these models in Chapter 4. The query flow graph which is a similar structure to the model we build with ant colony optimisation was enriched with clickthrough information. This is done

by observing the number of documents clicked after reformulating the query to adjust the weights on the corresponding edges. We also show a more elaborate use of click information where we consider clicks as an implicit source of relevance judgements which can be used for evaluating the retrieval performance of the underlying IR systems for each query in the log. These metrics can then be easily incorporated in the model as they are estimated for each query using aggregated clicks observed over time.

Can these adaptive domain models be applied for IR tasks?

The major application of our adaptive models that we studied in this thesis is *query recommendation*. We performed extensive evaluation to systematically explore the effectiveness of our adaptive models in providing query recommendation over time and compare their performances to established baselines. First we devised an evaluation methodology, AutoEval, that relies entirely on the search logs to assess the performance of the adaptive models, built from the same logs, to generate relevant query recommendations over time. We validated the results obtained by AutoEval with a corresponding user study in Chapter 5. The evaluation assumes that better systems should be able to perform better in a real scenario so it works by comparing suggestions derived in the model by reformulations actually observed in the logs. It also enables us to take the temporal aspect into account as it gives scores to the model to subsequent periods over time.

In Chapter 6, we performed extensive in-vitro evaluations with AutoEval to study our adaptive models using mainly the log data collected over a period of around 4 years from our reference system which is the University of Essex search engine. Also the Open University logs were further used for parts of the experiments. Our easily reproducible experiments quantitatively assessed our adaptive models allowing us to draw useful conclusions about how they perform over time and to understand the effect of their different parameters. As a summary of these conclusions, the ant colony optimisation outperforms Nootropia and the 'Association Rules' baseline model that all use query flows for query recommendation. We

show that automatically forgetting association over time which is the process of evaporating pheromone trails in ant colonies has a positive effect on the quality of query recommendation, however in our reference system if the forgetting is done on a high pace, i.e. forgetting quickly, this would harm the performance of query recommendation. The side effect of the evaporation process is the capability of the model to provide seasonally more relevant recommendations. The cross-season evaluation where we train a model on certain academic term and test on all the academic terms in the following year show that seasonally relevant recommendations are important in such an environment.

We also drew conclusions on building the enriched query flow graph where we interpret the user behaviour after reformulating a query to build the graph by observing the number of documents clicked. We found that incorporating the click data improves the performance for query recommendation, in particular boosting reformulations which result in a single click significantly improves the performance while somewhat surprisingly completely eliminating queries which result in no click downgraded performance.

In Chapter 7, we show another application of adaptive domain models in IR problems. We proposed to use these models to aid retrieval systems interpret query reformulations and previous user interactions in a user search session. Using the TREC Session Track which investigates this problem on Web search engines we show how adaptive domain models can be applied for the session retrieval problem. Generally, our approach is to implicitly derive query expansions from the adaptive models that better represent the user information needs throughout the session. Obtaining search logs from commercial Web search engines is not easy for academic research, hence we used anchor logs to simulate query logs. This would eliminate the temporal aspect of the logs which is necessary for building the adaptive models we presented in Chapters 3 and 4. Therefore we used the 'Association Rules' model, which is a cut-down version of our models, as an alternative. We show how query expansions relevant to the entire user session can be derived from the model. The results proved that this approach

is effective as it outperforms a simple baseline and in some cases all other systems submitted in the TREC Session Track. We did not limit ourselves to anchor logs as we also presented the 'Search Shortcuts' model on search logs from the MSN Web search engines. The 'Search Shortcuts' model was originally developed for query recommendation and has proven to be effective for rare or unseen queries. We devised an approach for employing 'Search Shortcuts' to derive relevant query expansions for the user session. The experiments show that they can improve the retrieval performance over a baseline system. We also studied the effect of different parameters of the recommendation process on the session retrieval performance.

8.2. Contributions

We summarise the contributions we made in this thesis as follows:

1. We explored two biologically inspired algorithms to build adaptive domain models from query flows in search logs and extensively evaluated these models for query recommendation. We also identified the effects of different parameters on their behaviour.

2. We introduced an intuitive, yet powerful, extension to a state-of-the-art model for query recommendation. The query flow graph was extended by interpreting the number of clicks observed after a query reformulation to assign different weights on the edges for each click behaviour. We show that the added value from the clicks has a positive effect on the quality of the query recommendation provided by the model.

3. Based on a principle that a better system should perform better in a real world scenario as estimated by a quantitative outcome, we devised a methodology for evaluating query recommender system over time. The methodology is controlled, deterministic and fully reproducible. It has the power of both identifying better models and also estimating the performance of the model *over time* which is an important matter for *adaptive* query recommendation systems.

4. We contributed a number of solutions to the session retrieval problem. Our adaptive models were used to model user information needs throughout the session mainly by deriving relevant query expansions from the model. Our solutions provided significant improvement over baseline systems and also they were among the top performing systems in the TREC Session Track.

8.3. The Wider Picture

In the thesis we presented the theoretical foundation of building a number of adaptive domain models and systematically evaluated their application for different IR tasks. Our work goes beyond the material of this thesis as we have implemented these models for real world search engines and applied them in digital libraries. Here we present two examples of the wider scope of the research conducted for producing this thesis.

8.3.1. SunnyAberdeen

SunnyAberdeen is a Web-based application that provides search and adaptation services for local websites or intranets (Albakour et al., 2011d). Figure 8.1 illustrates the logical components of the adaptive search framework. The framework is built as a *Web application* which can be used a standalone system or integrated as a Web service in other applications.

The web application has three main engines:

- **The Search Engine**: The search engine is a wrapper around an open source search engine that can index documents and produce search results for a user query. It is responsible for retrieving a ranked list of documents for a user query.
- **The Log Engine**: The log engine is responsible for logging all user interactions with the search framework. Whenever a new session starts, all the interactions of the user

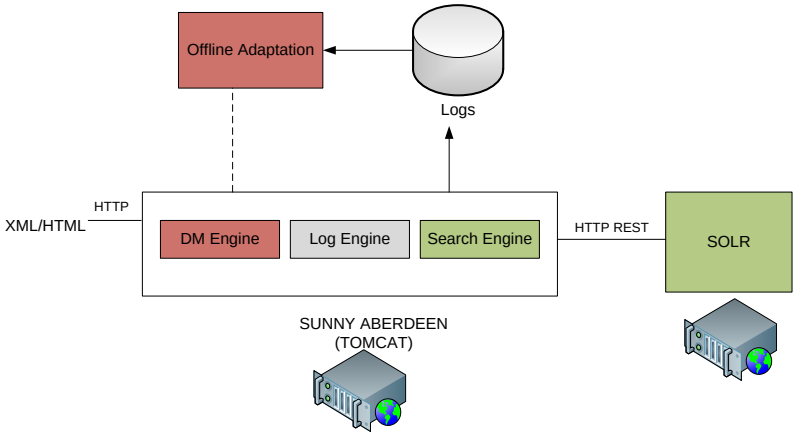


Figure 8.1.: Architecture

within that session are logged. This includes the queries that had been issued with their timestamps, the documents that have been retrieved and interactions with the various visual components and their timestamps.

- **The Domain Model Engine**: The domain model engine is responsible for maintaining and adapting a knowledge structure (the domain model) that reflects the document collection and the user community interest. It is consulted to produce query recommendations and to visualise these recommendations. Triggered periodically, an offline process consumes the logs recorded by the logging engine to update the domain model. This is where the adaptive models we introduced in Chapters 3, 4 are implemented which is the added value in this search framework over a standard one. It automatically updates the domain model from the user community interests that can aid future users to the right path to find information in addition to assisting them in browsing and navigation.

Figure 8.2 shows a snapshot of the user interface of the search framework. Alongside the search results displayed as a standard web search engine results, a number of interactive features are presented to the user. This includes a list of query recommendations, faceted browsing filters and a tag cloud of query refinements generated using the domain model adapted from previous logs of user interaction. Although not displayed in this screenshot, other interfaces can be instantiated from the framework such as an interactive connected graph of query suggestions.

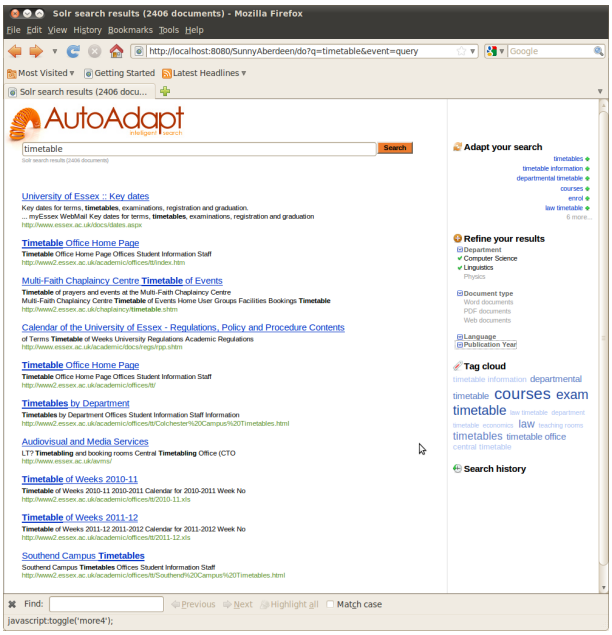


Figure 8.2.: A screenshot of the search framework

SunnyAberdeen is currently running as a live search engine for the University of Essex, and it is planned to be released to as open source software which gives more potential for extensions and contributions from the wider industrial and academic community.

8.3.2. Digital Libraries

In this thesis, we did most of our experiments to build and evaluate adaptive models using mainly the search logs from our University logs. However we have not limited ourselves to this domain, we also investigated building adaptive models using the log data collection of The European Library (TEL) (Albakour and Kruschwitz, 2011; Kruschwitz et al., 2011). Our findings there confirm the conclusions we have about the superiority of the ACO model over the baseline. However we found that TEL queries are particularly sparse. Whereas the top 20 most frequently submitted queries on our reference system make up as much as 15% of the entire query corpus, we observe that in a TEL log file of a comparable size to the university log the top 20 queries only cover about 2% of all queries submitted which make them more similar

to Web queries than intranet queries (Spink and Jansen, 2004). The difference of course is that Web logs are magnitudes larger which means that the actual count of even less frequent Web queries may still be large. Learning on the TEL logs is therefore particularly challenging. This was reflected in the low MRR scores we obtained in the automatic evaluation we conducted with AutoEval and in a corresponding user study using crowdsourcing. The scores were lower than those obtained for the same models using the university logs. The sparsity of the data and the open nature of the library domain makes it harder to learn useful query recommendations from the logs. The implication is that an effective learning algorithm will either have to rely on large log files or will have to exploit the logs much more effectively than what is needed when extracting relations from intranet or Web search logs.

8.4. Directions for Future Work

There is much room for future work either in building better adaptive models or in evaluating these models for IR applications. Here we discuss some ideas for potential future work. We have actually made progress in implementing some of these ideas.

The evaluation in Chapter 6 concluded that Nootropia's performance in providing relevant query recommendation outperformed the baseline but not the ACO model. This is promising, especially if we consider that it is a model developed originally for a different problem (information filtering). We also found out that its performance is mainly due to the weight of queries (nodes), rather than the weight of links. The ACO model does not have the notion of query weights. This triggers the investigation of hybrid approaches that combine the power of the two models, i.e., models that combine the link identification and weighting of ACO, with Nootropia's algorithm for learning node weights.

The evaluation experiments conducted on the enriched query flow graph only consider variants of the model by applying a combination of co-efficient factors for each of the click

behaviours. One area we will investigate is to automatically optimise these co-efficient factors. An extension of that work will then also allow us to look at incorporating signals other that the number of clicks after reformulation. This could include the time spent on viewing the documents, the scrolling behaviour and other signals. We will consider building a machine learning model which can be trained on actual search log data taking as features the aforementioned signals to optimise the graph weighting function. We have already collected training data for this purpose by collecting few thousands of query reformulations form the actual logs and asking experts of the domain to provide a binary judgement on whether each reformulation is useful or not.

The appeal of an automated evaluation framework is that we can re-run experiments and explore a large search space without any user intervention. The shortcoming is that any automated evaluation makes some simplifying assumptions, and end users will ultimately need to be involved to assess the real impact of the query recommendation suggestions being employed. We see our evaluation of our adaptive models for query recommendation as a first step in assessing what methods are promising and select those that promise the highest impact. Therefore we propose that the next step of evaluation is to conduct real-time experiments where we can assess the recommendation models by means of A/B testing, a common benchmarking technique widely used to make marketing decisions (Kohavi et al., 2007), on the live search engine. Under this setup each time a user submits a query, a model is selected randomly to provide query recommendations and various signals such as the take-up of recommendations can then be used to assess the recommendation models.

Another avenue for building adaptive models which we have not investigated in this thesis is using the search logs in combination with domain models extracted from the actual content of the document collection. The hypothesis here is that the integration of the two sources of knowledge would improve the model as they can be complementary to each other. We have already made progress in that direction. In (Adeyanju et al., 2012a), we propose to adapt the

concept subsumption hierarchy model originally introduced in (Sanderson and Croft, 1999) with query reformulations observed in the search logs. Two adaptation approaches using query logs with and without click information are compared. We evaluate the concept hierarchy models (static and adapted versions) built from the collections of two academic institutions and compare them with a the query flow graph model, built from the same logs. Our findings show that the adapted concept hierarchy model significantly outperforms its static version and the query flow graph when tested over a period of time on data (documents and search logs) from two institutions' intranets.

Bibliography

I. Adeyanju, D. Song, M-D. Albakour, U. Kruschwitz, A. De Roeck, and M. Fasli. Adaptation of the Concept Hierarchy model with search logs for query recommendation on Intranets. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012)*, 2012a. To appear.

I. Adeyanju, D. Song., F. Maria Nardini, M-D. Albakour, and U. Kruschwitz. RGU-ISTI-Essex at the TREC 2011 Session Track. In *Proceedings of the 20th Text Retrieval Conference (TREC 2011)*. NIST, 2012b.

G. Adomavicius and Y.O. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):896–911, 2012.

G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005. ISSN 1041-4347.

E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 19–26. ACM, 2006.

F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT 2005)*, pages 955–962. Association for Computational Linguistics, 2005.

M. Al Hasan, N. Parikh, G. Singh, and N. Sundaresan. Query Suggestion for E-Commerce Sites. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM 2011)*, pages 765–774, Hong Kong, 2011.

M-D. Albakour and U. Kruschwitz. University of Essex at LogCLEF 2011: Studying Query Refinement. In *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

M-D. Albakour, U. Kruschwitz, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Enriching Query Flow Graphs with Click Information. In *Proceedings of the 7th Asia Information Retrieval Societies Conference (AIRS 2011)*, pages 193–204, 2011a.

M-D. Albakour, U. Kruschwitz, N. Nanas, D. Song, M. Fasli, and A. De Roeck. Exploring Ant Colony Optimisation for Adaptive Interactive Search. In *Proceedings of the 3rd International Conference on the Theory of Information Retrieval (ICTIR 2011)*, Lecture Notes in Computer Science, pages 213–224, Bertinoro, 2011b. Springer.

M-D. Albakour, U. Kruschwitz, J. Niu, and M. Fasli. University of Essex at the TREC 2010 Session Track. In *Proceedings of the 19th Text Retrieval Conference (TREC 2010)*. NIST, 2011c.

M-D. Albakour, D. Lungley, and U. Kruschwitz. An adaptive search framework for intranets. In *Report of the symposium "Lernen, Wissen, Adaptivität" of the GI special interest groups KDML, IR and WM (LWA2011)*, pages 219–220, 2011d.

M-D. Albakour, N. Nanas, U. Kruschwitz, M. Fasli, Y. Kim, D. Song, and A. De Roeck. AutoEval: An Evaluation Methodology for Evaluating Query Suggestions Using Query Logs. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR*

2011), volume 6611 of *Lecture Notes in Computer Science*, pages 605–610, Dublin, 2011e. Springer.

M-D. Albakour, U. Kruschwitz, N. Nanas, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Analysis of Query Reformulations in a Search Engine of a Local Web Site. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*, volume 7224 of *Lecture Notes in Computer Science*, pages 517–521. Springer, 2012a.

M-D. Albakour, U. Kruschwitz, B. Neville, D. Lungely, M. Fasli, and N. Nanas. University of Essex at the TREC 2011 Session Track. In *Proceedings of the 20th Text Retrieval Conference (TREC 2011)*. NIST, 2012b.

M-D. Albakour, F. Maria Nardini, I. Adeyanju, and U. Kruschwitz. Studying Search Shortcuts in a Query Log to Improve Retrieval Over Query Sessions. In *Proceedings of the ECIR'12 workshop on Information Retrieval over Query Sessions (SIR 2012)*, 2012c.

A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 161–170, 2010.

P. Anick. Using Terminological Feedback for Web Search Refinement - A Log-based Study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 88–95, Toronto, Canada, 2003.

R. Aris. *Mathematical Modelling Techniques (Dover Books on Computer Science)*. Dover Publications, 1995. ISBN 0486681319.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition)*. Addison-Wesley Professional, 2011. ISBN 0321416910.

R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proceeding of the 13th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*

(SIGKDD 2007), pages 76–85, San Jose, California, 2007.

P. Bailey, A. P. de Vries, N. Craswell, and I. Soboroff. Overview of the TREC 2007 Enterprise Track. In *Proceedings of the 16th Text Retrieval Conference (TREC 2007)*, 2007.

R. Baraglia, F. Cacheda, V. Carneiro, D. Fernandez, V. Formoso, R. Perego, and F. Silvestri. Search shortcuts: a new approach to the recommendation of queries. In *Proceedings of the 3rd ACM conference on Recommender Systems (RecSys 2009)*, pages 77–84. ACM, 2009.

R. Baraglia, F. Maria Nardini, C. Castillo, R. Perego, D. Donato, and F. Silvestri. The Effects of Time on Query Flow Graph-based Models for Query Suggestion. In *Proceedings of the 9th RIAO Conference Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO 2010)*, pages 182–189, 2010.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly Analysis of a Very Large Topically Categorized Web Query Log. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 321–328, Sheffield, 2004.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, and D. Grossman. Temporal Analysis of a Very Large Topically Categorized Web Query Log. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(2):166–178, January 2007.

N. J. Belkin. Some(what) grand challenges for information retrieval. *SIGIR Forum*, 42(1): 47–54, 2008.

O. Ben-Yitzhak, N. Golbandi, N. Nadav, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining (WSDM 2008)*, pages 33–44, Palo Alto, 2008.

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 5:34–43, May 2001.

S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)*, pages 795–804, 2011.

M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1):92–128, 2005.

P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proceeding of the 17th ACM conference on Information and Knowledge Management (CIKM 2008)*, pages 609–618. ACM, 2008.

P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proceedings of the WSDM'09 workshop on Web Search Click Data (WSCD 2009)*, pages 56–63, 2009.

I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 515–522, Geneva, 2010.

E. Bortnikov, P. Donmez, A. Kagian, and R. Lempel. Modeling Transactional Queries via Templates. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*, volume 7224 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2012.

K. Bradley and B. Smyth. Improving recommendation diversity. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, 2001.

D. J. Brenes and D. Gayo-Avello. Automatic detection of navigational queries according to behavioural characteristics. In *LWA*, pages 41–48, 2008.

S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, pages 107–117, Brisbane, 1998.

D. Broccolo, O. Frieder, F. Maria Nardini, R. Perego, and F. Silvestri. Incremental algorithms for effective and efficient query recommendation. In *Proceedings of the 17th International Symposium on String Processing and Information Retrieval (SPIRE 2010)*, pages 13–24, 2010.

D. Broccolo, L. Marcon, F. Maria Nardini, R. Perego, and F. Silvestri. Generating suggestions for queries in the long tail with an inverted index. *Information Processing and Management*, 48(2):326–339, Mar 2012.

A. Z. Broder and A. C. Ciccolo. Towards the next generation of enterprise search technology. *IBM Systems Journal*, 43(3):451–454, July 2004.

P. Buitelaar and P. Cimiano. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, volume 167 of *Frontiers in Artificial Intelligence and Applications Series*. IOS Press, 2008.

P. Buitelaar, P. Cimiano, and B. Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications Series*. IOS Press, 2005.

G. Capannini, F. Maria Nardini, R. Perego, and F. Silvestri. Efficient diversification of web search results. *Proceedings of the VLDB Endowment*, 4(7):451–459, 2011.

B. Carterette and R. Jones. Evaluating Search Engines by Modeling the Relationship Between Relevance and Clicks. In *Advances in Neural Information Processing Systems 20*, pages 217–224. MIT Press, Cambridge, MA, 2008.

S. Ceccato. *Linguistic Analysis and Programming for Mechanical Translation (Mechanical Translation and Thought)*. Gordon and Breach, 1961.

M. Clark, Y. Kim, U. Kruschwitz, D. Song, M-D. Albakour, S. Dignum, U. Cervino Beresi, M. Fasli, and A. De Roeck. Automatically Structuring Domain Knowledge from Text: an Overview of Current Research. *Information Processing and Management*, 48(3):552–568, 2012.

C. W. Cleverdon. Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. Technical report, College of Aeronautics, 1962.

C. W. Cleverdon. The cranfield tests on index language devices. *ASLIB Proceedings*, 19(6): 173–194, 1967.

A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In *European Conference on Artificial Life*, pages 134–142, 1991.

G. V. Cormack, M. D. Smucker, and C. L. Clarke. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Information Retrieval*, 14:441–465, 2011.

N. Craswell and M. Szummer. Random Walks on the Click Graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 239–246, 2007.

N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC 2005 Enterprise Track. In *Proceedings of the 14th Text Retrieval Conference (TREC 2005)*, 2005.

N. Craswell, D. Fetterly, M. Najork, S. Robertson, and E. Yilmaz. Microsoft Research at TREC 2009: Web and relevance feedback tracks. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009)*. NIST, 2009.

W. B. Croft and D. D. Lewis. An Approach to Natural Language Processing for Document Retrieval. In *Proceedings of the 10th Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval (SIGIR 1987), pages 26–32, 1987.

M. Cuadros and G. Rigau. Quality assessment of large scale knowledge resources. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 534–541, Sydney, Australia, 2006. ACM.

S. Cucerzan and R. W. White. Query suggestion based on user landing pages. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 875–876. ACM, 2007.

J. Cullen and A. Bryman. The knowledge acquisition bottleneck: time for reassessment? *Expert Systems*, 5(3):216–225, 1988.

V. Dang and W. B. Croft. Query reformulation using anchor text. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 41–50. ACM, 2010.

G. Di Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

S. Dignum, U. Kruschwitz, M. Fasli, Y. Kim, D. Song, U. Cervino, and A. De Roeck. Incorporating Seasonality into Search Suggestions Derived from Intranet Query Logs. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence (WI 2010)*, pages 425–430, Toronto, 2010.

L. Doyle. Semantic road maps for literature searchers. *Journal of the ACM (JACM)*, 8(4): 553–578, 1961.

S. Dumais, T. Joachims, K. Bharat, and A. Weigend. SIGIR 2003 workshop report: implicit measures of user interests and preferences. *SIGIR Forum*, 37(2):50–54, 2003.

E. N. Efthimiadis. Query Expansion. In *Annual Review of Information Systems and Technology (ARIST)*, volume 31, pages 121–187. Information Today, 1996.

N. Eiron and K. S. McCurley. Analysis of anchor text for web search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 459–460. ACM, 2003.

D. W. Embley and B. Thalheim. *Handbook of Conceptual Modeling*. Springer, 1st edition edition, 2011.

R. Fagin, R. Kumar, Kevin S. McCurley, J. Novak, D. Sivakumar, J. A. Tomlin, and D. P. Williamson. Searching the workplace web. In *Proceedings of the 12th International World Wide Web Conference (WWW12)*, pages 366–375, 2003.

S. Feldman and C. Sherman. The high cost of not finding information. Technical Report 29127, IDC, 2001.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou. Ontology change: classification and survey. *The Knowledge Engineering Review*, 23(2):117–152, 2008.

B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *Proceedings of the 1st Latin American Web Congress*, pages 66–71, 2003.

B. M. Fonseca, P. B. Golgher, E. S. de Moura, B. Pôssas, and N. Ziviani. Discovering search engine related queries using association rules. *Journal of Web Engineering*, 2(4):215–227, 2004.

S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS)*, 23(2):147–168, April 2005.

N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems (TOIS)*, 7(3):183–204, July 1989.

W. Gao, C. Niu, J-Y. Nie, M. Zhou, K-F. Wong, and H-W. Hon. Exploiting query logs for cross-lingual query suggestions. *ACM Transactions on Information Systems (TOIS)*, 28(2), 2010.

D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Information Sciences*, 179:1822–1843, May 2009. ISSN 0020-0255.

A. Göker and D. He. Analysing web search logs to determine session boundaries for user-oriented learning. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)*, pages 319–322. Springer, 2000.

G. Golovchinsky, P. Qvarfordt, and J. Pickens. Collaborative Information Seeking. *IEEE Computer*, 42(3):47–51, 2009.

G. Grefenstette. Use of Syntactic Context to Produce Term Association Lists for Text Retrieval. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1992)*, pages 89–97. ACM, 1992.

H. Gürkök, M. Karamuftuoglu, and M. Schaal. A Graph Based Approach to Estimating Lexical Cohesion. In *Proceedings of the 2nd international Symposium on information interaction in Context (IIiX 2008)*, pages 35–43. ACM, 2008.

M. Harvey, M. Baillie, I. Ruthven, and M. James Carman. Tripartite Hidden Topic Models for Personalised Tag Suggestion. In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR 2010)*, pages 432–443, 2010.

A. Hassan, R. Jones, and K. L. Klinkner. Beyond dcg: user behavior as a predictor of a successful search. In *Proceedings of WSDM'10*, pages 221–230, 2010.

D. Hawking. Challenges in Enterprise Search. In *Proceedings of the 15th Australasian Database Conference (ADC 2004)*, pages 15–24, 2004.

D. Hawking. Enterprise Search. In R. Baeza-Yates and B. Ribeiro-Neto, editors, *Modern Information Retrieval*. Addison-Wesley Professional, 2011.

D. Hawking, F. Crimmins, N. Craswell, and T. Upstill. How Valuable is External Link Evidence When Searching Enterprise Webs? In *Proceedings of the 15th Australasian Database Conference (ADC 2004)*, pages 77–84, 2004.

Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, Ee-P. Lim, and H. Li. Web Query Recommendation via Sequential Query Prediction. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE 2009)*, pages 1443–1454. IEEE Computer Society, 2009.

M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 1992)*, volume 2, pages 539 – 545. ACL, 1992.

Marti A. Hearst. *Chapter 5: Presentation of Search Results*. Cambridge University Press, 2009. ISBN 0521113792.

D. Hiemstra and C. Hauff. Mirex: Mapreduce information retrieval experiments. Technical Report TR-CTIT-10-15, Centre for Telematics and Information Technology University of Twente, Enschede, April 2010.

W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIG conference on Human factors in computing systems (CHI 1995)*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

K. Hofmann, S. Whiteson, and M. de Rijke. Balancing Exploration and Exploitation in Learning to Rank Online. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR 2011)*, pages 251–263. Springer, 2011.

L. Hollink, G. Schreiber, and B. Wielinga. Patterns of semantic relations to improve image content search. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(3): 195–203, 2007.

E. Hovy, Z. Kozareva, and E. Riloff. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, volume 2, pages 948–957. ACL, 2009.

M. Hsu, M. Tsai, and H. Chen. Query Expansion with ConceptNet and WordNet: An Intrinsic Comparison. In *Proceedings of the 3rd Asia Information Retrieval Symposium (AIRS 2006)*, pages 1–13. Springer-Verlag, 2006.

J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceeding of the 18th ACM conference on Information and Knowledge Management (CIKM 2009)*. ACM, 2009.

B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000.

B. J. Jansen, A. Spink, C. Blakely, and S. Koshman. Defining a session on Web search engines. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(6): 862–871, April 2007.

B. J. Jansen, A. Spink, and I. Taksa, editors. *Handbook of Research on Web Log Analysis*. IGI, 2008.

K. Järvelin, S. L. Price, L. M. L. Delcambre, and M. Lykke Nielsen. Discounted Cumulated Gain Based Evaluation of Multiple-Query IR Sessions. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR 2008)*, pages 4–15. Springer, 2008.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceeding of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD 2002)*, pages 133–142, 2002.

T. Joachims. Evaluating Retrieval Performance using Clickthrough Data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*, pages 79–96. Physica/Springer Verlag, 2003.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 154–161. ACM, 2005.

T. Joachims, L. A. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

H. Joho, C. Coverson, M. Sanderson, and M. Beaulieu. Hierarchical presentation of expansion terms. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, pages 645–649. ACM, 2002.

H. Joho, M. Sanderson, and M. Beaulieu. A Study of User Interaction with a Concept-Based Interactive Query Expansion Support Tool. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR 2004)*, pages 42–56. Springer, 2004.

K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceeding of the 17th ACM conference on Information and Knowledge Management (CIKM 2008)*, pages 699–708. ACM, 2008.

R. Jones, B. Rey, and O. Madani. Generating query substitutions. In *Proceedings of the 15th International World Wide Web Conference (WWW15)*, pages 387–396, 2006.

J. S. Justeson and S. M. Katz. Technical Terminology Some Linguistic Properties and an Algorithm for Identification in Text. *Natural Language Engineering*, 1(1):9–27, 1995.

L. Pack Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

M. Kamvar and S. Baluja. Query suggestions for mobile search: understanding usage patterns. In *Proceedings of the 26th annual SIGCHI conference on Human factors in computing systems (CHI 2008)*, pages 1013–1016, 2008.

E. Kanoulas, P. Clough, B. Carterette, and M. Sanderson. Session Track at TREC 2010. In *Proceedings of the SIGIR'10 Workshop on the Automated Evaluation of Interactive Information (SimInt 2010)*, Geneva, Switzerland, 2010.

E. Kanoulas, B. Carterette, P. Clough, and M. Sanderson. Session Track 2010 Overview. In *Proceedings of the 20th Text Retrieval Conference (TREC 2011)*, 2011.

E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Session Track 2011 Overview. In *Proceedings of the 19th Text Retrieval Conference (TREC 2010)*, 2012.

M. P. Kato, T. Sakai, and K. Tanaka. Structured query suggestion for specialization and parallel movement: effect on search behaviors. In *Proceedings of the 21st international conference on World Wide Web (WWW21)*, pages 389–398, 2012.

D. Kelly. Methods for evaluating interactive information retrieval systems with users. *Found. Trends Inf. Retr.*, 3(1-2):1–224, January 2009. ISSN 1554-0669. doi: 10.1561/1500000012.

D. Kelly and N. J. Belkin. Reading time, scrolling and interaction: Exploring implicit sources of user preferences for relevance feedback during interactive information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 408–409. ACM, 2001.

D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 377–384, 2004.

D. Kelly, K. Gyllstrom, and E. W. Bailey. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 371–378. ACM, 2009.

R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceeding of the 13th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD 2007)*, pages 959–967, 2007.

M. Koolen and J. Kamps. The importance of anchor text for ad hoc search revisited. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 122–129. ACM, 2010.

R. Kraft and J. Zien. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*, Proceedings of the 13th International World Wide Web Conference (WWW13), pages 666–674, 2004.

U. Kruschwitz. An adaptable search system for collections of partially structured documents. *IEEE Intelligent Systems*, 18(4):44–52, July/August 2003.

U. Kruschwitz. *Intelligent Document Retrieval: Exploiting Markup Structure*, volume 17 of *the Information Retrieval series*. Springer, 2005.

U. Kruschwitz, N. Webb, and R. F. E. Sutcliffe. Query Log Analysis for Adaptive Dialogue-Driven Search. In J. Jansen, A. Spink, and I. Taksa, editors, *Handbook of Research on Web Log Analysis*, pages 389–416. IGI, Hershey, PA, 2008.

U. Kruschwitz, M-D. Albakour, J. Niu, J. Leveling, N. Nanas, Y. Kim, D. Song, M. Fasli, and A. De Roeck. Moving towards Adaptive Search in Digital Libraries. In *Advanced Language Technologies for Digital Libraries*, volume 6699 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2011.

N.K. Lathia. *Evaluating collaborative filtering over time*. PhD thesis, University College London, 2010.

D. Lenat, M. Prakash, and M. Shepherd. Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI magazine*, 6(4):65–85, 1985.

C. Lin and E. Hovy. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, volume 1, pages 495–501. ACL, 2000.

C. Liu, J. Gwizdka, J. Liu, T. Xu, and N. J. Belkin. Analysis and evaluation of query reformulations in different task types. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem*, 2010.

H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceeding of the 17th ACM conference on Information and Knowledge Management (CIKM 2008)*, pages 709–718, 2008.

Y. Maarek. The Surprising Role of Users in Web Search, 2012. Invited talk, April 3, the 34th European Conference on Information Retrieval (ECIR 2012), Barcelona, Spain.

A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

A. Maedche and S. Staab. Ontology Learning. In *Handbook on Ontologies*, chapter 9, pages 173–190. Springer Verlag, 2004.

A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. In *Proceedings of the 12th International World Wide Web Conference (WWW12)*, pages 439–448. ACM, 2003.

G. Marchionini. Human-information interaction research and development. *Library and Information Science Research*, 30(3):165–174, 2008.

G. Marchionini and B. Brunk. Towards a general relation browser: A GUI for information architects. *Journal of Digital Information*, 4(1), 2003.

K. Markey. Twenty-five years of end-user searching, Part 1: Research findings. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(8):1071–1081, June 2007.

D. Martens, M. De Backer, J. Vanthienen, M. Snoeck, and B. Baesens. Classification with Ant Colony Optimization. *IEEE Transactions on Evolutionary Computation*, 11:651–665, 2007.

Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and Knowledge Management (CIKM 2008)*, pages 469–478, 2008.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, volume 2, pages 1003–1011. ACL, 2009.

R. Mukherjee and J. Mao. Enterprise search: Tough stuff. *Queue*, 2(2):36–46, April 2004. ISSN 1542-7730.

N. Nanas. *Towards Nootropia: A Non-Linear Approach to Adaptive Document Filtering*. PhD thesis, Knowledge Media Institute, The Open University, UK, 2003.

N. Nanas, V. Uren, and A. De Roeck. Autopoiesis, the immune system and adaptive information filtering. *Natural Computing*, 8(2):387–427, 2009.

R. Navigli. Using cycles and quasi-cycles to disambiguate dictionary glosses. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 594–602, Athens, Greece, 2009. ACL.

R. Navigli and P. Velardi. An analysis of ontology-based query expansion strategies. In *Proceedings of the ECML Workshop on Adaptive Text Extraction and Mining (ATEM2003)*, pages 42–49, Cavtat Dubrovnik, Croatia, 2003.

R. Navigli and P. Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.

N. Nedjah and L. de Macedo Mourelle, editors. *Swarm Intelligent Systems*, volume 26 of *Studies in Computational Intelligence*. Springer, 2006.

P. Pantel and D. Lin. Discovering word senses from text. In *Proceeding of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD 2002)*, pages 613–619. ACM, 2002.

M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

M. Phillips. *Aspects of Text Structure: an investigation of the lexical organization of text*, volume 52 of *North-Holland Linguistic Series*. Elsevier, 1985.

R. M. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430, 1967.

F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD 2005)*, pages 239–248. ACM, 2005.

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceeding of the 17th ACM conference on Information and Knowledge Management (CIKM 2008)*. ACM, 2008.

R. Rao. From IR to Search, and Beyond. *Queue*, 2(3):66–73, May 2004.

S. Robertson. The probabilistic character of relevance. *Information Processing and Management*, 13(4):247–251, 1977.

S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. 1971.

I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 213–220. ACM, 2003.

I. Ruthven. Interactive information retrieval. *Annual Review of Information Science and Technology (ARIST)*, 42:43–92, 2008.

M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pages 206–213. ACM, 1999.

A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2002)*, pages 253–260. ACM, 2002. ISBN 1-58113-561-0.

H. Schütze. Automatic word sense discrimination. *Computational Linguistics - Special Issue on Word Sense Disambiguation*, 24(1):97–123, 1998.

C. Sherman. Why Enterprise Search will never be Google-y. *Enterprise Search Sourcebook*, pages 12–13, 2008.

C. Silverstein, M. Rauch Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

F. Silvestri. Mining Query Logs: Turning Search Usage Data into Knowledge. *Foundations and Trends in Information Retrieval*, 4:1–174, 2010.

B. Smyth, P. Briggs, M. Coyle, and M. O'Mahony. Google Shared. A Case-Study in Social Search. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2009)*, pages 283–294. Springer, 2009.

R. Snow, D. Jurafsky, and A. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 801–808. ACL, 2006.

I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 66–73. ACM, 2001.

I. Soboroff, A. P. de Vries, and N. Craswell. Overview of the trec 2006 enterprise track. In *Proceedings of the 15th Text Retrieval Conference (TREC 2006)*, 2006.

K. Socha, M. Sampels, and M. Manfrin. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In *Applications of Evolutionary Computing*, volume 2611 of *Lecture Notes in Computer Science*, pages 334–345. Springer Berlin / Heidelberg, 2003.

J. F. Sowa. Conceptual Graphs. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, chapter 5, pages 213–237. Elsevier, 2008.

A. Spink and B. J. Jansen. *Web Search: Public Searching of the Web*, volume 6 of *The Information Science and Knowledge Management Series*. Kluwer, 2004.

A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. *Journal of the American Society for Information Science and Technology (JASIST)*, 52(3):226–234, February 2001.

T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, 2004.

I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th International World Wide Web Conference (WWW20)*, pages 47–56, 2011.

Y. Y. Tang, C. D. Yan, and C. Y. Suen. Document processing for automatic knowledge acquisition. *IEEE Transactions on Knowledge and Data Engineering*, 6:3–21, 1994.

A. Toumouth, A. Lehireche, D. Widdows, and M. Malki. Adapting WordNet to the medical domain using lexicosyntactic patterns in the ohsumed corpus. In *Proceedings of the 2006 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2006)*, pages 1029–1036. ACL, 2006.

D. J. Tunkelang. *Faceted search*. Morgan & Claypool Publishers, 2009.

T. Upstill, N. Craswell, and D. Hawking. Query-independent evidence in home page finding. *ACM Transactions on Information Systems (TOIS)*, 21(3):286–313, 2003.

D. Vallet, M. Fernández, and P. Castells. An ontology-based information retrieval model. *The Semantic Web: Research and Applications*, pages 455–470, 2005.

L. van der Plas and J. Tiedemann. Using lexico-semantic information for query expansion in passage retrieval for question answering. In *Proceedings of the COLING Workshop on Information Retrieval for Question Answering (IRQA 2008)*, pages 50–57. ACL, 2008.

P. Velardi, P. Fabriani, and M. Missikoff. Using text processing techniques to automatically enrich a domain ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS 2001)*, pages 270–284. ACM, 2001.

E. Vorhees and D. Harman. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005. ISBN 0262220733.

C. Wagner. Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Information Resources Management Journal*, 19(1):70–83, 2006.

P. Wang, M. W. Berry, and Y. Yang. Mining Longitudinal Web Queries: Trends and Patterns. *Journal of the American Society for Information Science and Technology (JASIST)*, 54(8): 743–758, June 2003.

J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10^{th} International World Wide Web Conference (WWW10)*, pages 162–168. ACM, 2001.

M. White. *Making Search Work: Implementing Web, Intranet and Enterprise Search*. Information Today, 2007.

R. W. White and J. Huang. Assessing the scenic route: measuring the value of search trails in web logs. In *Proceedings of the 33^{rd} International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 587–594. ACM, 2010.

R. W. White and R. A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.

R. W. White and I. Ruthven. A Study of Interface Support Mechanisms for Interactive Information Retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 57(7):933–948, 2006.

R. W. White, M. Bilenko, and S. Cucerzan. Studying the Use of Popular Destinations to Enhance Web Search Interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 159–166. ACM, 2007.

D. Widdows. *Geometry and Meaning*. CSLI Lecture Notes, 2004.

D. Widdows and B. Dorow. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, volume 1, pages 1–7. ACL, 2002.

D. Widdows, S. Cederberg, and B. Dorow. Visualisation Techniques for Analysing Meaning. In *Proceedings of the 30th International Conference on Text, Speech, and Dialogue (TSD2002)*, pages 107–114, 2002.

Y. Wilks and C. Brewster. *Natural Language Processing as a Foundation of the Semantic Web*. Foundations and Trends in Information Retrieval, 2006.

W. A. Woods. What's in a Link: Foundations for Semantic Networks. In D. Bobrow and A. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, 1975.

X. Yuan and N. J. Belkin. Supporting multiple information-seeking strategies in a single system framework. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 247–254. ACM, 2007.

X. Zhu, J. Guo, and X. Cheng. Recommending diverse and relevant queries with a manifold ranking based approach. In *Proceedings of the SIGIR'10 workshop on Query Representation and Understanding*, 2010.