Learning to Select for Information Retrieval

Jie Peng

Department of Computing Science
Faculty of Information and Mathematical Sciences

University of Glasgow

A thesis submitted for the degree of

Doctor of Philosophy

June 2010

Abstract

The effective ranking of documents in search engines is based on various document features, such as the frequency of the query terms in each document, the length, or the authoritativeness of each document. In order to obtain a better retrieval performance, instead of using a single or a few features, there is a growing trend to create a ranking function by applying a learning to rank technique on a large set of features. Learning to rank techniques aim to generate an effective document ranking function by combining a large number of document features. Different ranking functions can be generated by using different learning to rank techniques or on different document feature sets. While the generated ranking function may be uniformly applied to all queries, several studies have shown that different ranking functions favour different queries, and that the retrieval performance can be significantly enhanced if an appropriate ranking function is selected for each individual query.

This thesis proposes Learning to Select (LTS), a novel framework that selectively applies an appropriate ranking function on a per-query basis, regardless of the given query's type and the number of candidate ranking functions. In the learning to select framework, the effectiveness of a ranking function for an unseen query is estimated from the available neighbouring training queries. The proposed framework employs a classification technique (e.g. k-nearest neighbour) to identify neighbouring training queries for an unseen query by using a query feature. In particular, a divergence measure (e.g. Jensen-Shannon), which determines the extent to which a document ranking function alters the scores of an initial ranking of documents for a given query, is proposed for use as a query feature. The ranking function which performs the best on the identified training query set is then chosen for the unseen query.

The proposed framework is thoroughly evaluated on two different TREC retrieval tasks (namely, Web search and adhoc search tasks) and on two large standard LETOR feature sets, which contain as many as 64 document features, deriving conclusions concerning the key components of LTS, namely the query feature and the identification of neighbouring queries components. Two different types of experiments are conducted. The first one is to select an appropriate ranking function from a number of candidate ranking functions. The second one is to select multiple appropriate document features from a number of candidate document features, for building a ranking function. Experimental results show that our proposed LTS framework is effective in both selecting an appropriate ranking function and selecting multiple appropriate document features, on a per-query basis. In addition, the retrieval performance is further enhanced when increasing the number of candidates, suggesting the robustness of the learning to select framework.

This thesis also demonstrates how the LTS framework can be deployed to other search applications. These applications include the selective integration of a query independent feature into a document weighting scheme (e.g. BM25), the selective estimation of the relative importance of different query aspects in a search diversification task (the goal of the task is to retrieve a ranked list of documents that provides a maximum coverage for a given query, while avoiding excessive redundancy), and the selective application of an appropriate resource for expanding and enriching a given query for document search within an enterprise. The effectiveness of the LTS framework is observed across these search applications, and on different collections, including a large scale Web collection that contains over 50 million documents. This suggests the generality of the proposed learning to select framework.

The main contributions of this thesis are the introduction of the LTS framework and the proposed use of divergence measures as query features for identifying similar queries. In addition, this thesis draws insights from a large set of experiments, involving four different standard collections, four different search tasks and large document feature sets. This illustrates the effectiveness, robustness and generality of the LTS framework in tackling various retrieval applications.

Acknowledgements

I owe my deepest gratitude to all those who gave the immense support through the journey of my PhD.

Firstly, I am heartily thankful to my supervisor, Iadh Ounis. His wide knowledge and his logical way of thinking have been of great value for me. His help, stimulating suggestions and encouragement helped me throughout my research and the writing up of this thesis

My warm thanks are due to Craig Macdonald and Rodrygo L. T. Santos for reading parts of this thesis, and for their comments on various drafts of this thesis.

I would also like to thank those with whom I had a wonderful time in the past and at present. These include Vassilis Plachouras, Ben He, Christina Lioma, Alasdair Gray, Stuart Chalmers, and Richard Mccreadie.

Finally, I would like to thank my family, especially my wife Ming Chen. Their care and love made it possible for me to complete this work. They have always encouraged me to follow my dreams.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Motivation 3

1.3 Thesis Statement 4

1.4 Contributions 5

1.5 Origins of the Material 6

1.6 Thesis Outline 7

2 Information Retrieval **10**

2.1 Introduction 10

2.2 Indexing 11

2.2.1 Tokenisation 11

2.2.2 Stop Words Removal 12

2.2.3 Stemming 12

2.2.4 Index Data Structures 13

2.3 Matching 14

2.3.1 Best Match Weighting Models 16

2.3.2 Language Modelling 18

2.3.3 Divergence From Randomness 20

2.4 Query Expansion and Collection Enrichment 24

2.4.1 Query Expansion 24

2.4.2 Collection Enrichment 27

2.4.3 Query Performance Predictors 28

2.5 Term Dependency 36

2.5.1 Term Dependency via Markov Random Fields 38

2.5.2 Term Dependency in DFR . . . 42
2.6 Query Independent Features . . . 44
2.6.1 Link-based Query Independent Features . . . 45
2.6.2 URL-based Query Independent Features . . . 46
2.6.3 Other Query Independent Features . . . 48
2.6.4 The Integration of Query Independent Feature . . . 49
2.7 Evaluation . . . 50
2.8 Summary . . . 53

3 Learning to Rank and Selective Retrieval Approaches 54
3.1 Introduction . . . 54
3.2 Learning to Rank . . . 55
3.2.1 The Pointwise Approach . . . 55
3.2.2 The Pairwise Approach . . . 58
3.2.3 The Listwise Approach . . . 63
3.2.4 Summary . . . 70
3.3 Query Dependent Ranking . . . 70
3.3.1 The KNN Online Algorithm . . . 71
3.3.2 The KNN Offline-1 Algorithm . . . 72
3.3.3 The KNN Offline-2 Algorithm . . . 73
3.4 Selective Collection Enrichment . . . 75
3.5 Selective Web Information Retrieval . . . 76
3.6 Query Type Prediction . . . 78
3.7 Summary . . . 79

4 Learning to Select 81
4.1 Introduction . . . 81
4.2 The Learning to Select Framework . . . 82
4.3 Selecting Multiple Document Features . . . 84
4.4 The Components of Learning to Select . . . 85
4.4.1 Query Features . . . 85
4.4.2 Identifying Neighbouring Queries . . . 91
4.4.3 Variants of Learning to Select . . . 96

4.5 Example of Learning to Select 96
4.6 Discussion 98
4.7 Summary 100

5 Experiments using Learning to Select 102
5.1 Introduction 102
5.2 Research Questions 103
5.3 Experimental Settings 103
5.3.1 Retrieval Tasks 104
5.3.2 Collections and Topics 104
5.3.3 Retrieval Baselines 106
5.3.4 Training Procedure 106
5.4 The Importance of Selective Application 108
5.5 Selecting A Single Ranking Function 108
5.5.1 The Effectiveness of Learning to Select 108
5.5.2 The Robustness of Learning to Select 112
5.6 Selecting Multiple Document Features 114
5.7 Summary 122

6 Learning to Select in Other Search Applications 123
6.1 Introduction 123
6.2 Selective Application of Query Independent Features 124
6.2.1 Research Questions 124
6.2.2 Experimental Settings 125
6.2.3 Experimental Results 127
6.2.4 Summary 132
6.3 Web Search Diversification 135
6.3.1 The xQuAD Framework 136
6.3.2 Research Questions 137
6.3.3 Experimental Settings 138
6.3.4 Experimental Results 139
6.3.5 Summary 144
6.4 Enterprise Document Search 145

6.4.1 Research Questions 146
6.4.2 Experimental Settings 147
6.4.3 Experimental Results 149
6.4.4 Summary 162
6.5 Summary 163

7 Conclusions and Future Work 164
7.1 Contributions and Conclusions 164
7.1.1 Contributions 164
7.1.2 Conclusions 167
7.2 Directions for Future Work 169

A Additional Tables 172

References 200

List of Figures

2.1 An example of the linkage between terms in a sentence, where stop words are bracketed. 38
2.2 An example of the Markov random fields model with a full independence assumption. 39
2.3 An example of the Markov random fields model with a sequential dependence assumption. 40
2.4 An example of the Markov random fields model with a full dependence assumption. 41
2.5 An illustration of the link structure of documents. 46
2.6 An illustration of the click distance feature. 49

3.1 An illustration of the SVM classifier. 58
3.2 An illustration of the procedure of the Ranking SVM approach during the training process. 62
3.3 An illustrative example of the ranking of four objects by adjusting the weight vector. 63
3.4 An illustration of the KNN Online algorithm for building a document ranking function. 72
3.5 An illustration of the KNN Offline-1 algorithm for building a document ranking function. 73
3.6 An illustration of the KNN Offline-2 algorithm for building a document ranking function. 74

4.1 Score distributions of the top 1000 retrieved documents, which are ranked according to their relevance scores in the base ranking function. 86
4.2 The distribution of the relative retrieval performance in MAP, with standard error bars, versus the estimated divergence score. 87
4.3 An illustration of the KNN algorithm for classifying objects. . . . 92
4.4 An illustration of the k-means algorithm for classifying objects. . 94

5.1 MAP versus the number of candidate ranking functions on the TREC 2003 dataset. 115
5.2 MAP versus the number of candidate ranking functions on the TREC 2004 dataset. 116
5.3 MAP versus the number of candidate ranking functions on the TREC 2007 dataset. 117
5.4 MAP versus the number of candidate ranking functions on the TREC 2008 dataset. 118

6.1 Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2007 enterprise document search using the PL2F weighting scheme. 158
6.2 Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2007 enterprise document search using the BM25F weighting scheme.159
6.3 Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2008 enterprise document search using the PL2F weighting scheme. 160
6.4 Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2008 enterprise document search using the BM25F weighting scheme.161

List of Tables

2.1 An example of the inverted index structure. We assume that a corpus only contains one document, which only has a single sentence, which is the same as our example sentence. 14
2.2 An illustrative example of the query expansion and collection enrichment algorithms. 28
2.3 Overview of the techniques employed by the predictors. Columns *Single* and *Co-occur* refer to the use of the statistics of single query terms and the co-occurrence of multiple query terms, respectively. 29
2.4 The proposed features for phrase identification. 37

3.1 The highest score in each column is highlighted in bold and scores that are statistically better than *RankingSVM*, *AdaRank*, and AFS are marked with $\star$, $*$, and †, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$). 70
3.2 The decision mechanism for the selective application of CE. $score_L$ and $score_E$ denote the predicted performance of a given query on the local and external resources, respectively. *local*, *external* and *disabled* in the column *Decision* indicate expanding the initial query on the *local* resource, *external* resource and disabling the expansion, respectively. 76
3.3 The description of the query features used in the classification process for query type detection. 78

4.1 An example of query feature scores and MAP evaluations for 10 training queries and 2 ranking functions. 96

4.2 The distance between each training query q_ϕ and the given test query q', which is an absolute value of the subtraction between the two queries' feature scores. 97
4.3 The ranking of training queries according to the computed query distance scores. 98

5.1 Salient properties of the .GOV corpus 104
5.2 The number of queries for the TREC 2003 & 2004 Web Track and TREC 2007 & 2008 Million Query Track. 105
5.3 For each dataset, the highest score in each column is highlighted in bold and scores that are statistically better than *RankingSVM*, *AdaRank*, and *AFS* are marked with $\star$, $*$, and $\dagger$, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$). 107
5.4 Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 3.0 TREC 2004 dataset. Results are the mean over 5 folds. 109
5.5 Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 3.0 TREC 2004 dataset. Results are the mean over 5 folds. 110
5.6 Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 4.0 TREC 2007 dataset. Results are the mean over 5 folds. 111
5.7 Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 4.0 TREC 2008 dataset. Results are the mean over 5 folds. 112
5.8 The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2003 dataset with different numbers of candidate document features. Results are the mean over 5 folds. 120
5.9 The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2004 dataset with different numbers of candidate document features. Results are the mean over 5 folds. 121

5.10 The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2007 dataset with different numbers of candidate document features. Results are the mean over 5 folds. 121
5.11 The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2008 dataset with different numbers of candidate document features. Results are the mean over 5 folds. 121

6.1 Details of the number and percentage of topics associated to each topic type for TREC 2003 and TREC 2004 Web tracks. 125
6.2 The MAP upper bounds, highlighted in bold, which are achieved by the manual selective application of a query independent feature on the TREC 2003 and TREC 2004 datasets. 128
6.3 Evaluation of the learning to selective framework for selectively applying a query independent feature between PageRank (PR) and URL depth (UD). 130
6.4 Evaluation of the learning to selective framework for selectively applying a query independent feature between PageRank (PR) and click distance (CD). 130
6.5 Evaluation of the learning to selective framework for selectively applying a query independent feature between URL depth (UD) and click distance (CD). 131
6.6 Evaluation of the learning to selective framework for selectively applying a query independent feature among PageRank (PR), URL depth (UD), and click distance (CD). 133
6.7 Comparison between the best performing variant of learning to select and the oracle QTP method for the selective application of a query independent feature among PageRank, URL depth, and click distance. 133

6.8 The highest score based on each weighting scheme is highlighted in bold and scores that are statistically better than the corresponding xQuAD$_U$, xQuAD$_W$, and xQuAD$_C$ are marked with $*$, $\diamond$, and $\star$, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$). 140
6.9 Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the DPH weighting scheme. 141
6.10 Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the BM25 weighting scheme. 142
6.11 Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the language modelling weighting scheme. 143
6.12 Overview of the test collections for expanding a given query. . . . 148
6.13 The MAP and nDCG results for applying QE and CE systematically using the PL2F document weighting model, and Upper Bound highlighted in bold, which is achieved by the optimal selective application of CE. 150
6.14 The MAP and nDCG results for applying QE and CE systematically using the BM25F document weighting model, and Upper Bound highlighted in bold, which is achieved by the optimal selective application of CE. 151
6.15 Evaluation of the selective application of collection enrichment on the TREC 2007 enterprise document search task using the PL2F weighting scheme. 152
6.16 Evaluation of the selective application of collection enrichment on the TREC 2007 enterprise document search task using the BM25F weighting scheme. 153
6.17 Evaluation of the selective application of collection enrichment on the TREC 2008 enterprise document search task using the PL2F weighting scheme. 155

6.18 Evaluation of the selective application of collection enrichment on the TREC 2008 enterprise document search task using the BM25F weighting scheme. 156

A.1 Document features included in the LETOR 3.0 dataset. 174

A.2 Document features included in the LETOR 4.0 dataset. 175

Chapter 1

Introduction

1.1 Introduction

The effective ranking of documents in information retrieval (IR) systems is based on various document features, such as the frequency of query terms in each document and the length or authority score of each document. In general, document features can be categorised into two groups: query dependent and query independent. Query dependent document features are mainly based on the statistics of query terms in a document (Upstill et al., 2003), such as the frequency of query terms in the document or can be computed by more sophisticated models, e.g., BM25 document weighting model (Robertson et al., 1994). These query dependent features are computed for each issued query. The query independent document features, also known as document priors, relating perhaps to document content, linkage or usage, are computed before retrieval time and regardless of the query. They can be transformed into a static, per-document relevance weight for use in ranking (Craswell et al., 2005), e.g., PageRank (Brin & Page, 1998).

Broder (2002) identified that current Web search users' information needs, which are represented as queries, can be classified into three types: *informational*, whose intent is to acquire some information assumed to be present on one or more web pages; *navigational*, whose intent is to reach a particular site; and *transactional*, whose intent is to perform some web-mediated activity, such as online shopping. Different types of queries benefit differently from different document features. For example, Kraaij et al. (2002) showed that URL type is a very effective document feature for finding the homepage of an organisation and Peng et al. (2007) showed that informational queries usually benefit from the modelling of term dependency, or term co-occurrence.

In order to obtain a better retrieval performance, instead of using a single or a few document features, there is a growing trend to create a ranking function based on a large set of various document features (Geng et al., 2008; Metzler, 2007; Xu & Li, 2007). This is illustrated by the emergence of the *learning to rank* field in information retrieval, where a ranking function is usually generated based on a large number of document features. Learning to rank techniques generate a ranking function by assigning a weight to each document feature, then use this generated ranking function to estimate the relevance score for each document. Several different learning to rank techniques have been proposed in the literature (Burges et al., 2005; Cao et al., 2007; Herbrich et al., 2000; Nallapati, 2004; Xu & Li, 2007). They mainly differ in terms of the loss function used to guide the learning process (Liu, 2009).

By using different learning to rank techniques, different ranking functions can be generated based on different document feature sets. Most of the current approaches usually use a single ranking function to assign a relevance score to each document for all given queries, i.e., they *systematically apply* the same ranking function to all queries. In contrast, the process of selecting an appropriate ranking function from a number of *candidate* ranking functions and applying it to a given query is called the *selective application* of ranking functions.

Several studies have shown that retrieval performance can be significantly enhanced if an appropriate ranking function is applied for each individual query (Kamps et al., 2004; Peng & Ounis, 2009; Peng et al., 2009; Plachouras, 2006; Plachouras & Ounis, 2004). In this thesis, we propose to selectively apply an appropriate ranking function from a large number of candidate ranking functions for each given query, regardless of the given query's type. In addition, in the context of the learning to rank paradigm, we also propose to selectively apply multiple appropriate document features for building a ranking function, on a per-query basis.

The remainder of the introduction describes the motivation for the work in this thesis (Section 1.2). We present the thesis statement (Section 1.3) and its contributions (Section 1.4). We also list the origins of the material (Section 1.5), and close this chapter with an overview of the structure for the remainder of the thesis (Section 1.6).

1.2 Motivation

To build an effective information retrieval system that satisfies all types of users' information needs, most of the current approaches tend to build a ranking function on a large number of different document features. Several learning to rank techniques have been proposed to build such a ranking function, and different ranking functions can be generated by using different learning to rank techniques on different document feature sets. These generated ranking functions are usually systematically applied to all queries. However, many studies have shown that different queries benefit differently from each ranking function (Kamps et al., 2004; Peng & Ounis, 2009; Peng et al., 2009; Plachouras, 2006; Plachouras & Ounis, 2004) and that the retrieval performance can be significantly enhanced if an appropriate ranking function is used for each individual query. Several selective retrieval approaches have been proposed in the literature (Geng et al., 2008; Peng et al., 2009; Plachouras & Ounis, 2004; Song et al., 2004; Yang et al., 2004), which aim to enhance the retrieval performance by applying different retrieval strategies to different queries. However, these approaches have various limitations and shortcomings, as described below.

Geng et al. (2008) proposed to apply a specific ranking function for each given query. This specific ranking function is generated by applying a learning to rank technique on a set of neighbouring training queries. However, this approach is only investigated using a single learning to rank technique and using a fixed set of document features. Its effectiveness is not clear when several different learning to rank techniques are used and when the number of document features is varied.

Peng et al. (2009) proposed a decision mechanism to decide whether a given query should be expanded using an internal or an external resource, on a per-query basis. The decision mechanism is based on the performance (e.g. "good" or "bad") of a given query on the internal and external resources. Such performance is obtained by using a query performance predictor, e.g., query scope (He & Ounis, 2006). However, this approach may not be applicable to the selective application of a ranking function. Indeed, the predictors mainly rely on the statistics of the collection, which are invariant to changes in the ranking function.

Plachouras (2006) proposed a method to selectively apply an appropriate retrieval approach for a given query, which is based on a Bayesian decision mechanism. Features such as the occurrence of query terms in the documents were used to determine the applicability of the retrieval approaches. The retrieval performance obtained using this approach only improved slightly over the systematic application of a retrieval approach and actually decreased when more than two candidate retrieval approaches are used.

Song et al. (2004) and Yang et al. (2004) tried to apply different retrieval approaches for different query types (named page, homepage, and topic distillation). To achieve this, they proposed several different techniques (e.g. the use of a linguistic classifier) to detect the query type. However, the obtained accuracy of the query type prediction is not high (Craswell & Hawking, 2004) and queries of the same type may benefit from having different retrieval approaches applied (Peng & Ounis, 2009).

This thesis investigates a novel framework, which is effective in selecting an appropriate ranking function from a number of candidate ranking functions for a given query, regardless of the given query's type as well as the number of candidate ranking functions/document features. In particular, this framework should not only be able to select an appropriate ranking function, but should also be able to select multiple appropriate document features for building a ranking function, on a per-query basis.

1.3 Thesis Statement

The statement of this thesis is that an appropriate ranking function can be successfully selected from a number of candidate ranking functions for each given query, regardless of the given query's type and the number of candidate ranking functions/document features. This is investigated in the context of a framework, called Learning to Select (LTS), where the effectiveness of a ranking function for a given query is estimated based on its performance on the already seen neighbouring queries. In the learning to select framework, if the neighbouring queries of a given query are successfully identified, then the ranking function that performs the best on the identified neighbouring queries is selected for the given query. The learning to select framework comprises two components: a component for *identifying the neighbouring* queries and a component called *query feature*. The identifying neighbouring queries component employs a classification technique (e.g. k-nearest neighbour (Cover & Hart, 1967)) to identify a certain number of neighbouring queries for a given query, by using a query feature which is a representative of the characteristics of a given query. The query feature component is used to build such a query feature. In particular, two divergence measures (i.e. Kullback-Leibler (Kullback, 1997) and Jensen-Shannon (Lin, 1991)), used to determine the extent to which a document ranking function alters the scores of an initial ranking of documents, are proposed as query features in the thesis.

1.4 Contributions

The main contributions of this thesis are the following. A novel framework, called learning to select (LTS), is introduced for selectively applying an appropriate ranking function on a per-query basis, and regardless of the given query's type and the number of candidate ranking functions/document features. In this framework, the effectiveness of a ranking function for an unseen query is estimated based on its retrieval performance on the query's already seen neighbouring queries. In addition, we identify the main components of the learning to select framework: identifying neighbouring queries and query feature. In particular, we propose to use divergence measures as query features for identifying neighbouring queries.

Moreover, the differences between the learning to select framework and the previously proposed selective retrieval approaches (described in Section 1.2) are discussed. In the course of the thesis, two key research questions concerning the learning to select framework are addressed, namely how to effectively select an appropriate ranking function, and how to effectively select a set of appropriate document features for building a ranking function, on a per-query basis.

Furthermore, we thoroughly evaluate the effectiveness and robustness of the learning to select (LTS) framework on two different retrieval tasks, namely Web search and ad hoc search, and on two large standard document feature sets, which contain as many as 64 document features. Moreover, we study in detail how each component of the learning to select framework affects retrieval performance.

Finally, to show the generality of the LTS framework, in Chapter 6, we investigate how the LTS framework can be used in other search applications. These applications include the selective application of a query independent feature. This allows the effective integration of an appropriate query independent document feature into a document weighting scheme on a per-query basis. Therefore, different document features are applied to the queries that are more likely to benefit from these features; the selective estimation of the relative importance of different query aspects in a search diversification task. This allows the ranked list of documents to provide a complete coverage of different interpretations for an ambiguous query. In particular, a Web collection that contains over 50 million documents is used in this evaluation; and the selective application of an appropriate resource for expanding a given query, called *selective query expansion*, for document search within an enterprise. This alleviates the mismatch problem between query terms and the intranet documents. The mismatch problem is severe in an enterprise, because usually only a small number of people tend to create documents according to autocratic guidelines, reflecting the enterprise policies.

1.5 Origins of the Material

The material that form parts of this thesis have found their origins in various papers that I have published during the course of my PhD research. In particular:

- The learning to select framework as defined in Chapter 4 is based on work published in (Peng & Ounis, 2009) (ECIR 2009). The outline of the experiments in Chapter 5 is somewhat similar to those published in ECIR 2010 (Peng et al., 2010).

- The use of divergence measures as query features, as defined in Section 4.4.1, are based on work initially published in SIGIR 2008 (Peng et al., 2008).

- The use of bins to identify neighbouring queries, as defined in Section 4.4.2, is based on work published in (Peng et al., 2007) (RIAO 2007) and (Peng & Ounis, 2007) (ECIR 2007).
- The use of query performance predictors to conduct a selective external query expansion (Section 3.4) is based on work initially published in ICTIR 2009 (Peng et al., 2009). The experiments on choosing an appropriate resource for expanding a given query (Section 6.4) are based on work published in (Peng et al., 2009) (CIKM 2009).
- The modelling of term dependency introduced in Section 2.5.2 and used in Section 6.2 is based on work published in SIGIR 2007 (Peng et al., 2007).

1.6 Thesis Outline

The remainder of this thesis is organised as follows:

- Chapter 2 presents various document features on which this thesis relies, including both query dependent and query independent document features. The query dependent document features described in this chapter are derived from various document weighting models (BM25, language modelling and Divergence From Randomness), query expansion approaches, and the modelling of the term dependency. In addition, various query independent features (including PageRank, URL type and click distance) are described.
- Chapter 3 provides an overview of several different learning to rank techniques, which are used to generate a ranking function by assigning a weight to each document feature. These generated ranking functions are usually systematically applied to all queries, and ignore the fact that different queries benefit differently from each ranking function. Based on a literature survey, we introduce several existing selective retrieval approaches. However, these approaches are either not applicable to the selective application of a ranking function, or are not able to effectively select an appropriate ranking function from a large number of candidate ranking functions, or are totally dependent on the identified query type which ignores the fact

that the same type of queries may benefit differently from the same ranking function.

- Chapter 4 introduces the learning to select framework for selectively applying an appropriate ranking function on a per-query basis. First, it presents the general idea and the detailed algorithm of the proposed learning to select framework. Next, it explains how to use the learning to select framework to select multiple document features for building a ranking function. Following this, the chapter provides the description of each component of the learning to select framework and presents an example illustrating how the learning to select framework operates. Finally, the chapter closes with a detailed discussion on the learning to select framework in comparison with the existing approaches in the literature.

- Chapter 5 presents the evaluation of the proposed learning to select framework. The evaluation is conducted on two different retrieval tasks (i.e. Web search and ad hoc search tasks) and on two different standard document feature sets (i.e. LETOR 3.0 and LETOR 4.0). The document feature sets used contain as many as 64 different document features, including query independent and query independent features. Two different evaluations of the learning to select framework are conducted in this chapter, including selecting an appropriate ranking function from a number of candidate ranking functions and selecting a set of appropriate document features from a number of candidate document features for building a ranking function. Moreover, each component of the LTS framework is thoroughly studied.

- Chapter 6 explores the deployment of the learning to select framework in other search applications. In particular, we describe an experiment to determine if the learning to select framework can be effectively applied to integrate an appropriate query independent feature into a document weighting scheme. Moreover, we examine the effectiveness of our proposed learning to select framework in choosing an appropriate resource (i.e. document collection) for expanding an initial query. Lastly, we test how effective the learning to select framework is, by applying it to the xQuAD search diversification framework, in which an appropriate sub-query importance estimator is selectively applied.

- Chapter 7 closes this thesis with the contributions and the conclusions drawn from this work, as well as possible directions of future work across the investigated search tasks.

Chapter 2

Information Retrieval

2.1 Introduction

Information Retrieval (IR) aims at modelling, designing, and implementing systems able to provide fast and effective content-based access to large amounts of information (Baeza-Yates, 2003). The aim of an IR system is to estimate the relevance of information items, such as text documents, images and video, to a user's *information need*. Such information need is represented in the form of a *query*, which usually corresponds to a bag of words. Users are only interested in the information items that are *relevant* to their information need. The primary goal of an IR system is to retrieve all the information items that are relevant to a user query while retrieving as few non-relevant items as possible (Baeza-Yates & Ribeiro-Neto, 1999). Furthermore, the retrieved information items should be ranked from the most relevant to the least relevant.

In contrast to data retrieval, which aims to retrieve all objects that satisfy a clearly defined condition (van Rijsbergen, 1979), information retrieval usually deals with natural language text which is not always well structured and could be semantically ambiguous (Baeza-Yates & Ribeiro-Neto, 1999). Therefore, there tend to be non-relevant items(s) which could be ignored in IR. The information retrieval field has been evolving quickly in recent years but some core techniques are still widely used in most IR systems, such as the indexing process, which will be described in Section 2.2.

This chapter mainly provides an overview of various document features that this thesis relies on, including both query dependent and query independent document features. The query dependent document features described in this chapter include: IR models for matching information items with queries (Section 2.3), the query expansion and collection enrichment techniques which are used to reformulate users' information need (Section 2.4), and the term dependency technique, which aims to model the dependency between terms (Section 2.5). Apart from these query dependent document features, we also introduce various query independent document features, also called as document priors, such as PageRank, in Section 2.6. At last, the evaluation of IR systems is described in Section 2.7.

2.2 Indexing

For IR systems, in order to efficiently judge whether the documents from a corpus match a given query, a pre-process called indexing is usually applied. In general, the objective of indexing is to extract representatives (e.g. terms) for each document and to store them in a specific data structure (e.g. inverted file), which provides an efficient access to these document representatives.

To better explain the indexing process, we use the following sentence as an example, taken from (Austen, 1813):

"**It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife.**"

The indexing process includes several steps, which are described in the following of this section.

2.2.1 Tokenisation

The first stage of the indexing process is typically known as tokenisation (Manning et al., 2008), which is the task of chopping documents into tokens based on the boundary between document tokens, such as whitespaces. In addition, at this stage, all characters contained in the tokens are often lower-cased and all punctuations are removed. After tokenisation, the above example sentence can be viewed as:

it is a truth universally acknowledged that a single man in possession of a good fortune must be in want of a wife

2.2.2 Stop Words Removal

Luhn (1957) pointed out that the frequency of a term within a document can be a good discriminator of its significance in the document. In addition, there are many extremely frequent terms (e.g. “the”) that appear in almost all documents of a corpus. These terms are called *stop words*, which bring little value for the purpose of representing the content of documents and are normally filtered out from the list of potential indexing terms during the indexing process (Baeza-Yates & Ribeiro-Neto, 1999). Removing the stop words allows also the reduction of the size of the generated document index.

Articles, prepositions, and conjunctions are natural candidates for building such a list of stop words. Moreover, the stop words list can be extended by determining the most frequent or least informative terms in the document collection (Lo et al., 2005). As an example, a list of 425 stop words is illustrated in (Frakes & Baeza-Yates, 1992). However, the general trend of IR systems over time moved from the standard use of a large list of stop words (200 - 300 terms) towards the use of a smaller list of stop words (7 - 12 terms), by exploiting the statistics of language (Manning et al., 2008). After applying stop words removal to our example sentence, the text is reduced to the following:

> **truth universally acknowledged single man possession fortune want wife**

2.2.3 Stemming

Often, a user specifies a term (e.g. *democracy*) in a query when only a variant of this term (e.g. *democratic*) is contained in a relevant document. Hence, it would be beneficial for retrieval if documents containing variants of the query term were also considered. Plurals, gerund forms, and past tense suffixes are examples of syntactical variations which prevent a perfect match between a query term and a respective document term (Baeza-Yates & Ribeiro-Neto, 1999). This problem can be alleviated by applying stemming, which replaces a term with its stem, so that different grammatical forms of terms are represented in a common base form. A *stem* is the portion of a word which is obtained after chopping off its affixes.

An example of a stem is the word *employ*, which is the stem of *employment*, *employer*, *employee*, *employed* and *employable*.

There are several different suffix removal algorithms for the English language, such as the one-pass Lovins stemmer (Lovins, 1968) which is the first stemming algorithm, the best known Porter's stemmer (Porter, 1980) and the newer technique - the Paice/Husk stemmer (Paice, 1990). Among these stemming algorithms, Porter's stemmer is the most widely used because of its simplicity and elegance. In general, Porter's stemmer consists of five phases of suffix removal. Within each phase, there are various conventions to select rules, such as selecting the rule from each rule group that applies to the longest suffix (Manning et al., 2008). By applying Porter's stemming algorithm, our example sentence is transformed as follows:

truth univers acknowledg singl man possess fortun want wife

It is easy to note that some words are unchanged (e.g. "truth" and "wife"), some are chopped to their root (e.g. "possess") while some are transformed into non-English words (e.g. "univers" and "singl").

2.2.4 Index Data Structures

To enable efficient access to document representatives, a suitable data structure is necessary. The most widely used data structure is the inverted index (Frakes & Baeza-Yates, 1992), which is a word-oriented mechanism. In general, the inverted index structure contains two components: the *vocabulary* and the *occurrences* (Baeza-Yates & Ribeiro-Neto, 1999). The vocabulary is a set of all different terms extracted from the corpus by the above steps. The occurrences store each vocabulary term's statistics in each document, such as term frequency and term position (see Table 2.1 for an example).

The size of the inverted index can be very large. This can be reduced by applying appropriate compression techniques, through the way of encoding the integers, which are normally used to represent document identifiers and term frequencies, with fewer bits or bytes. The bit-wise Elias-Gamma and Elias-Unary encodings (Elias, 1975) are commonly used for compressing document identifiers and term frequencies, respectively (Witten et al., 1999). Both of them are

vocabulary	occurrences	
	term frequency	term position
truth	$< 1, 1 >$	$< 1, 1 >$
univers	$< 1, 1 >$	$< 1, 2 >$
acknowledg	$< 1, 1 >$	$< 1, 3 >$
singl	$< 1, 1 >$	$< 1, 4 >$
man	$< 1, 1 >$	$< 1, 5 >$
possess	$< 1, 1 >$	$< 1, 6 >$
fortun	$< 1, 1 >$	$< 1, 7 >$
want	$< 1, 1 >$	$< 1, 8 >$
wife	$< 1, 1 >$	$< 1, 9 >$

Table 2.1: An example of the inverted index structure. We assume that a corpus only contains one document, which only has a single sentence, which is the same as our example sentence.

parameter-free and achieve an acceptable compression and fast decoding. In addition, it is possible to obtain a higher compression rate and a faster decoding by applying parameterised techniques, such as Golomb codes (Golomb, 1966). However, the choice of an appropriate parameter has a significant impact on the compression rate when using a parameterised model. Some other compression techniques operate on bytes instead of bits, with the aim of exploiting the optimised capability of hardware to handle bytes (Williams & Zobel, 1999). Apart from saving disk space usage, deploying a compressed index brings an additional benefit to the retrieval speed for an IR system (Scholer et al., 2002).

2.3 Matching

With a given query, an ideal IR system should only return relevant documents and rank these documents in decreasing order of relevance. The relevance of a document to a given query can be estimated by various IR models, such as the Boolean model, which is one of the oldest IR models. It is based on set theory and Boolean algebra (Spärck Jones & Willett, 1997). Queries in the Boolean model are formulated as Boolean expressions. As an example, with a given Boolean query `birthday AND cake (NOT card)`, the Boolean model would retrieve all documents that contain both the terms `birthday` and `cake` but not the term `card`. In addition, due to its binary decision criterion (i.e., when using the Boolean model, a document is predicted to be either relevant or non-relevant), the retrieved documents are returned to the user as a set without any ranking. The lack of a ranking of the result set has been one of the major drawbacks of the Boolean model (Salton et al., 1983*a*).

In contrast to the Boolean model, several other IR models have been proposed to estimate the relevance of a document. One such model is the vector space model (Salton & McGill, 1986*a*), where both queries and documents are represented as vectors in the same space. Each vector dimension corresponds to a separate term and the number of dimensions of the vector space corresponds to the number of distinct terms in the corpus. In particular, the term occurrences are usually weighted by the TF-IDF weighting scheme, given as follows:

$$score(d, Q) = \sum_{t \in Q} tf \cdot log_2 \frac{N}{N_t} \tag{2.1}$$

where tf is the number of occurrences of the term t from query Q in document d. N is the total number of documents in the collection, and N_t is the number of documents in which t occurs. The TF component is motivated by the premise that the more frequently a term occurs in a document, the more important the term is for this document (Salton & McGill, 1986*b*). The component $log_2 \frac{N}{N_t}$ is called the *inverse document frequency* (IDF). The IDF component is used to diminish the importance of terms have very high frequency and to increase the importance of terms have a rare occurrence in the collection, which is also called *term specificity* (Spärck Jones, 1972). A term with high IDF (namely low N_t) is more useful than a term with low IDF. Moreover, Robertson & Spärck Jones (1988) proposed several methods for measuring a term's specificity. Furthermore, the retrieved documents are ranked according to their similarity to the query, which can be computed by various measures, such as the cosine similarity.

Another classical retrieval model is the probabilistic model (Robertson & Spärck Jones, 1988), which is popular due to its effectiveness and strong theoretical foundations. Robertson (1977) assumed that the probability of relevance of a document to a query is independent of other documents, then posed the probability ranking principle (PRP), which states that:

> "If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

The remainder of the current section describes particular families of the probabilistic models. Section 2.3.1 describes the family of Best Match (BM) models. The language modelling approach for IR is presented in Section 2.3.2. Section 2.3.3 discusses the Divergence From Randomness (DFR) framework of information retrieval models.

2.3.1 Best Match Weighting Models

Starting from a basic probabilistic model (Robertson & Spärck Jones, 1988), the weight of a term t in a document is computed as follows:

$$w^{(1)} = log\frac{(r+0.5)/(R-r+0.5))}{(N_t-r+0.5)/(N-N_t-R+r+0.5)} \tag{2.2}$$

where R is the number of relevant documents, r is the number of relevant documents containing the term t, N_t is the number of documents where the term t appears and N is the number of documents contained in the collection. In addition, the above equation can be simplified when the relevance information is not available (Croft & Harper, 1997):

$$w^{(1)} = log\frac{N-N_t+0.5}{N_t+0.5} \tag{2.3}$$

$w^{(1)}$ is similar to the inverse document frequency (idf): $log\frac{N}{N_t}$.

However, the above equations only contain the IDF concept but not TF. Robertson et al. (1980) approached this problem by combining the probabilistic model with the 2-Poisson model. Harter (1975) extended the original 2-Poisson work with the hypothesis that a set of *specialty* terms occur randomly across the document collection while they occur relatively more densely in an *elite* set of documents. The elite set of documents would be the answer to a query that contains the specialty terms. However, this approach involves several parameters that cannot be set in a straight-forward manner. For this reason, Robertson & Walker (1994) approximated the 2-Poisson model of term frequencies with a simpler formula but with similar shapes and properties, given as follows:

$$w = \frac{tfn}{k_1 + tfn} w^{(1)} \tag{2.4}$$

where k_1 is a parameter that controls the saturation of tfn, which is computed as follows:

$$tfn = \frac{tf}{(1-b) + b \cdot \frac{l}{avg\_l}} \tag{2.5}$$

where l is the length of document in tokens, avg_l is the average length of all documents in the corpus, and b is the term frequency normalisation hyper-parameter.

Robertson et al. (1994) proposed the Best Match 25 (BM25) model, which is used in our experiments (Chapter 6), given as follows:

$$score(d,Q) = \textstyle\sum_{t \in Q} \left(\frac{(k_1+1)tfn}{k_1 + tfn} \cdot \frac{(k_3+1)qtf}{k_3 + qtf} \cdot w^{(1)}\right) \tag{2.6}$$

where k_3 is a parameter that controls the saturation of the term frequency in the query and qtf is the frequency of a term in the query.

Document structure (or fields), such as the title and the anchor text of incoming hyperlinks, have been shown to be effective in Web IR (Craswell & Hawking, 2004). Robertson et al. (2004) proposed the idea of normalising term frequency on a per-field basis and Zaragoza et al. (2004) introduced a field-based version of BM25, called BM25F (used in Chapter 6), which applies length normalisation and weighting of the fields independently.

$$score(d,Q) = \textstyle\sum_{t \in Q} \left(\frac{(k_1+1)tfn}{k_1 + tfn} \cdot \frac{(k_3+1)qtf}{k_3 + qtf} \cdot w^{(1)}\right) \tag{2.7}$$

The above BM25F' weighting function is the same as Equation (2.6). However, BM25F applies a per-field normalisation method to assign the normalised term frequency tfn:

$$tfn = \sum_f w_f \cdot tfn_f = \sum_f w_f \cdot \frac{tf_f}{(1-b_f) + b_f \cdot \frac{l_f}{avg\_l_f}} \tag{2.8}$$

where w_f is the weight of a field f, which reflects the relative contribution of a field to the document ranking. tfn_f is the normalised term frequency on field f. tf_f is the frequency of the query term in the field f. l_f is the number of tokens in the field f. avg_f is the average length of field f in the collection.

2.3.2 Language Modelling

The aforementioned 2-Poisson indexing based retrieval models have two assumptions: the first assumption is that the within-document term frequencies (i.e. tf) follow a mixture of two Poisson distributions; the second assumption is that the elite set of documents would be the answer to a query that contains the specialty terms. Ponte & Croft (1998) suggested that it is not necessary to make any parametric assumption when the actual data is available and proposed the application of a language modelling approach in information retrieval. The intuition is that users have a reasonable idea of which terms are likely to occur in documents of interest and these terms (query) can be used to distinguish the relevant documents from others in the collection.

In the language modelling approach, documents and queries are considered as a sequence of words and the probability of a document d being relevant to a given query Q can be formulated after applying Bayes' rule:

$$P(d|Q) = \frac{P(Q|d) \cdot P(d)}{P(Q)} \tag{2.9}$$

where $P(Q)$ can be safely ignored as it does not affect the ranking of documents, $P(d)$ is the prior probability of relevance of the document d. $P(Q|d)$ is the probability of generating the query Q given the document d. There are various ways to estimate this probability. In (Ponte & Croft, 1998), the probability of generating a query from a document language model is equivalent to the product of the probability of generating each of the query terms multiplied by the product of the probability of not generating the terms that do not appear in the query. However, in (Hiemstra, 1998), the probability of generating a query from a document language model is only based on the product of the probability of generating the query terms from the document language model, which is simpler, as it ignores the terms that do not occur in the query:

$$P(Q|d) = \prod_{t \in Q} P(t|d) \tag{2.10}$$

where $P(t|d)$ can be estimated with various models, such as:

$$P(t|d) = \frac{tf}{l} \tag{2.11}$$

where tf is the number of occurrences of the query term t in document d and l is the document length in tokens.

However, $P(t|d)$ could be zero if the term t does not appear in the document d. Ponte & Croft (1998) suggested that it is harsh to assign a zero probability to these terms as documents that do not contain a query term will not be retrieved. Instead, they proposed to supplement the document model with a collection model, which is the probability of a query term t occurring in the entire collection. This solution is also known as *smoothing*. Zhai & Lafferty (2004) studied the effectiveness of various smoothing techniques for language modelling in IR, such as Jelinek-Mercer, Dirichlet and Absolute discount. The experimental results show that not only is the retrieval performance generally sensitive to the smoothing parameters, but also the sensitivity pattern is affected by the query length, with performance being more sensitive to smoothing for long queries than for short queries.

In (Hiemstra, 2001), a language modelling approach with Jelinek-Mercer smoothing for ranking documents, which will be used in our experiments (Chapter 6), is given as follows:

$$\begin{aligned} score(d, Q) &= \prod_{t \in Q} P(t|d) \\ &\approx \sum_{t \in Q} log(1 + \frac{\lambda \cdot tf \cdot \#(token)}{(1-\lambda) \cdot TF \cdot l}) \end{aligned} \tag{2.12}$$

where λ is the hyper-parameter of the Jelinek-Mercer smoothing, which is between 0 and 1. tf is the number of occurrences of the query term t in the document d and $\#(token)$ is the number of tokens in the entire collection. TF is the number of occurrences of the query term t in the whole collection and l is the document length in tokens.

2.3.3 Divergence From Randomness

Amati (2003) proposed the Divergence From Randomness (DFR) framework for building probabilistic IR weighting models, which is a generalisation of Harter's 2-Poisson indexing model and based on the following idea:

> "The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word t in the document d."

In the DFR framework, the weight of a term t in a document d is a function of two probabilities:

$$w(t,d) = (1 - Prob_2(tf|E_t)) \cdot (-log_2 Prob_1(tf|Collection)) \tag{2.13}$$

where $-log_2 Prob_1(tf|Collection)$ is the probability that the term t appears with frequency tf in a document by chance, according to a given model of randomness, also know as the *randomness model* component; $1 - Prob_2(tf|E_t)$ corresponds to the information gain obtained by considering a term to be informative for a document, also known as the *aftereffect of sampling* component. E_t stands for the elite set of documents, which is already defined in Section 2.3.1. In addition, the term frequency tf can be normalised with respect to the length of the document, so that all documents are treated equally (Singhal et al., 1996). This is called the *document length normalisation* component. The three components will be detailed in the rest of this section.

2.3.3.1 Randomness model

In the *randomness model* component, $-log_2 Prob_1(tf|Collection)$ is higher if the probability that a term t occurs tf times is lower. A term is considered to be more informative when $-log_2 Prob_1(tf|D)$ is higher. If the occurrences of a term are distributed according to a binomial model, then the probability of observing tf occurrences of a term in a document is given by the probability of tf successes in a sequence of TF Bernoulli trials with N possible outcomes:

$$Prob_1(tf|Collection) = \binom{TF}{tf} \cdot p^{tf} \cdot q^{TF-tf} \tag{2.14}$$

where TF is the frequency of a term in a collection of N documents, $p = \frac{1}{N}$ and $q = 1 - p$.

There are various models that can be used as an approximation of the Bernoulli model, such as the Poisson model (denoted as P in the DFR framework). Assuming that the maximum likelihood estimator $\lambda = \frac{TF}{N}$ of the frequency of a term in this collection is low, or, in other words, $TF \ll N$, the Poisson distribution can be used to approximate the binomial model described above. In this case, the informative content of $Prob_1$ is given as follows:

$$-log_2 Prob_1(tf|Collection) = tf \cdot log_2 \frac{tf}{\lambda} + (\lambda - tf) \cdot log_2 e + 0.5 \cdot log_2(2\pi \cdot tf) \quad (2.15)$$

2.3.3.2 Aftereffect of sampling

In the DFR framework, $(1 - Prob_2(tf|E_t))$ takes into account the notion of aftereffect (Feller, 1968) of observing tf occurrences of t in a document. It corresponds to the information gain obtained by considering a term to be informative for the weighted document. If a term appears with a high frequency in a document, then it is almost certain that this term is informative for this document. Amati (2003) noted that the informative terms are rare in the collections but, in compensation, when they occur, their frequency is very high, which indicates the importance of these terms in the corresponding documents. In other words, it may happen that a sudden repetition of success of a rare event increases our expectation of a further success to almost certainty. Laplace's law of succession (denoted as L in the DFR framework) is one of the possible estimates of such an expectation.

$$1 - Prob_2(tf|E_t) = 1 - \frac{tf}{1 + tf} = \frac{1}{1 + tf} \quad (2.16)$$

An alternative model for computing $Prob_2(tf|E_t)$ is the Bernoulli model (denoted as B in the DFR framework), which is defined as the ratio of two binomial distributions:

$$1 - Prob_2(tf|E_t) = \frac{TF}{n \cdot (tf + 1)} \quad (2.17)$$

where n is the document frequency of the term in the document collection.

2.3.3.3 Document length normalisation

The length of the document simply corresponds to the number of indexed tokens and different documents usually have different document lengths. Before using Equation (2.13) to estimate the weight of a term in a document, the term frequency tf is usually normalised with repsect to the document length, so that all documents are treated equally. Amati (2003) assumed a decreasing density function of the normalised term frequency with respect to the document length and derived the following formula, which is called *normalisation 2*:

$$tfn = tf \cdot log_2(1 + c \cdot \frac{avg\_l}{l}) \tag{2.18}$$

where tfn is the normalised term frequency, l is the document length, avg_l is the average document length in the whole collection and c ($c > 0$) is a hyper-parameter that controls the normalisation applied to the term frequency with respect to the document length. If $c = 1$, then Equation (2.18) becomes:

$$tfn = tf \cdot log_2(1 + \frac{avg\_l}{l}) \tag{2.19}$$

which is called *normalisation 1* in the DFR framework (Amati, 2003).

2.3.3.4 Divergence From Randomness Weighting Models

A DFR document weighting model is generated from a combination of a randomness model for computing $-log_2 Prob_1(tf|Collection)$, an aftereffect model for computing $1 - Prob_2(tf|E_t)$, and a term frequency normalisation model. Take the PL2 weighting model as an example, which will be used in Chapter 6, the randomness model is the Poisson distribution (Equation (2.15)), the information gain is computed with the Laplace model (Equation (2.16)), and the term frequencies are adjusted using *normalisation 2* (Equation (2.18)), given as follows:

$$score(d, Q) = \sum_{t \in Q} \frac{1}{1 + tfn}(tfn \cdot log_2 \frac{tfn}{\lambda} + (\lambda - tfn) \cdot log_2 e + 0.5 \cdot log_2(2\pi \cdot tfn)) \tag{2.20}$$

DFR also generates a series of hyper-geometric models. The hyper-geometric distribution is a discrete probability distribution that describes the number of successes in a sequence of draws from a finite population without replacement. Amati et al. (2007) proposed a hyper-geometric model, called DPH (used in Section 6.3), which assigns the relevance score of a document d for a query Q as follows:

$$\begin{aligned} score(d,Q) = &\sum_{t\in Q} \frac{(1-F)^2}{tf+1} \cdot \big(tf \cdot \log_2(tf \cdot \frac{avg\_l}{l}\frac{N}{TF})) \\ &+ 0.5 \cdot \log_2(2\pi \cdot tf \cdot (1-F))\big) \end{aligned} \quad (2.21)$$

where F is given by tf/l, tf is the within-document frequency, and l is the document length in tokens. avg_l is the average document length in the collection, N is the number of documents in the collection, and TF is the term frequency in the collection. Note that DPH is a parameter-free model, and therefore requires no particular tuning. qtw is the query term weight and is given by qtf/qtf_{max}, where qtf is the query term frequency and qtf_{max} is the maximum query term frequency among all query terms.

Macdonald et al. (2005) introduced *Normalisation 2F* in the DFR framework for performing independent term frequency normalisation and weighting of fields, and proposed the PL2F model, which assigns the relevance score of a document d for a query Q as follows:

$$\begin{aligned} score(d,Q) \quad = \quad &\sum_{t\in Q} \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} \\ &+(\lambda - tfn) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big) \end{aligned} \quad (2.22)$$

where λ is the mean and variance of a Poisson distribution, given by $\lambda = F/N$; F is the frequency of the query term t in the whole collection, and N is the number of documents in the whole collection. qtf is the query term frequency; qtf_{max} is the maximum query term frequency among the query terms.

The above PL2F' weighting function is the same as Equation (2.20). However, in PL2F, tfn corresponds to the weighted sum of the normalised term frequencies tf_f for each used field f, known as *Normalisation 2F* (Macdonald et al., 2005):

$$tfn = \sum_f \left(w_f \cdot tf_f \cdot \log_2(1 + c_f \cdot \frac{avg\_l_f}{l_f}) \right), (c_f > 0) \quad (2.23)$$

where tf_f is the frequency of term t in field f of document d; l_f is the length in tokens of field f in document d, and avg_l_f is the average length of the field across all documents; c_f is a hyper-parameter for each field, which controls the term frequency normalisation; the importance of the term occurring in field f is controlled by the weight w_f.

2.4 Query Expansion and Collection Enrichment

Queries composed by the users are not always well structured and may be inadequate or incomplete representations of user's information need, as most queries are quite short and only contain 2 to 3 terms (Jansen et al., 1998; Silverstein et al., 1999). This could result in word mismatch between query and documents (Carpineto & Romano, 2004). The problem of word mismatch is fundamental to IR. The most commonly used approach to solve this problem is query expansion, which is the process of expanding the original query with additional terms with a similar meaning or some other statistical relationships.

2.4.1 Query Expansion

Voorhees (1994) investigated the expansion of short queries based on the lexical-semantic relations between terms. In her investigation, a lexical database, called WordNet (Miller et al., 1990), was employed. However, little evidence that the retrieval effectiveness could be enhanced by expanding the original query with the selected terms was shown. The selected terms were obtained based on their lexical relation with the original query terms.

Spärck Jones (1971) proposed to cluster terms based on their term co-occurrence in the documents and used the obtained clusters to expand the given queries. Following this idea, a number of similar approaches have been proposed and are either based on global analysis or local analysis. For the global analysis approaches, the term co-occurrence in documents is investigated in the whole corpus (Schütze & Pedersen, 1997; Xu & Croft, 1996). As for the local analysis approaches, only the top retrieved documents are involved with the term co-occurrence investigation (Attar & Fraenkel, 1977; Xu & Croft, 1996). Compared to the local analysis, the global analysis is more computationally expensive and less effective in retrieval performance (Xu & Croft, 1996).

Most of the current query expansion algorithms (Carpineto et al., 2001; Robertson, 1990; Rocchio, 1971*a*) follow a similar outline to Spärck Jones' original work in (Spärck Jones, 1971). In general, these algorithms contain the following three steps to conduct query expansion:

Step 1 For a given query, use an IR system to obtain a list of documents, which is also called as relevant feedback.

Step 2 Rank the terms that are contained in the relevant feedback using a specific term weighting model.

Step 3 The top ranked terms are added to the original query, and documents are retrieved by using the expanded query.

For the first and third steps, there are many document weighting models that can be used to obtain the relevant feedback and rank documents with the expanded query, such as language modelling (Equation (2.12)), BM25 (Equation (2.6)) and PL2 (Equation (2.20)). In particular, in the first step, there are three types of relevant feedback: explicit feedback, implicit feedback and pseudo-relevant feedback. Explicit feedback is obtained through user interaction (White et al., 2001). For a given query, the corresponding top retrieved documents are judged by the users using a binary (i.e. "relevant" or "irrelevant") or graded (e.g. "not relevant", "somewhat relevant", "relevant", or "highly relevant") relevance scale; Implicit feedback is obtained from users' behaviour instead of their explicit judgement (Kelly & Belkin, 2001). Such behaviour includes the duration of time spent reading a document, page browsing and scrolling actions. Lastly, pseudo-relevant feedback is obtained by assuming that the top retrieved documents are relevant. It has the advantage that no user interaction is required.

2.4.1.1 Term Weighting Models

As for the second step, several term weighting models have been proposed for ranking the terms contained in the relevant feedback (Amati, 2003; Lavrenko & Croft, 2001; Robertson, 1990). Here, we introduce two different term weighting models: the Bo1 term weighting model and the KL-based term weighting model, which have shown their effectiveness in several retrieval tasks (Amati, 2003; Hannah et al., 2007; McCreadie et al., 2009). The Bo1 and KL-based term weighting models are based on the Bose-Einstein statistics and the KL divergence measure, respectively. The basic idea of the two term weighting models is to measure the divergence of a term's distribution in a relevant feedback from its distribution in the whole collection. The higher this divergence is, the more likely the term is related to the topic of the query.

By using Pseudo-relevant feedback, for the Bo1 term weighting model, the weight $w(t)$ of a term t in the top T ranked documents is given by:

$$w(t) = tf_x \cdot \log_2 \frac{1+P_n}{P_n} + \log_2(1+P_n) \tag{2.24}$$

where T is usually ranges from 3 to 10 (Amati, 2003), TF is the frequency of the term t in the collection, N is the number of documents in the collection, P_n is $\frac{TF}{N}$, and tf_x is the frequency of the query term in the top T ranked documents

The terms with highest $w(t)$ scores from the top T ranked documents are extracted. The number of extracted terms $\#(term)$ is another parameter involved in the query expansion mechanism and is usually larger than T (Amati, 2003).

After expanding the original query with new query terms, a parameter-free function is used to determine the query term weight qtw for each query term, given as follows:

$$\begin{aligned} qtw &= \frac{qtf}{qtf_{max}} + \frac{w(t)}{\lim_{TF \to tf_x} w(t)} \\ &= TF_{max} \log_2 \frac{1+P_{n,max}}{P_{n,max}} + \log_2(1+P_{n,max}) \end{aligned} \tag{2.25}$$

where $\lim_{TF \to tf_x} w(t)$ is the upper bound of $w(t)$, $P_{n,max}$ is given by TF_{max}/N, and TF_{max} is the TF of the term with the maximum $w(t)$ in the top ranked documents. If a query term does not appear in the most informative terms from the top-ranked documents, its query term weight remains unchanged.

Another term weighting model which has been introduced is based on the KL divergence measure. Using the KL model, the weight of a term t in the feedback document set D is given by (Amati, 2003):

$$w(t) = p(t|D) \cdot \log_2 \frac{p(t|D)}{p(t|Coll)} \tag{2.26}$$

where $p(t|D) = tf_x/c(D)$ is the probability of observing the term t in the feedback document set D, tf_x is the frequency of the term t in the set D, $c(D)$ is the number of tokens in this set, $p(t|Coll) = TF/\#(token)$ is the probability of observing the term t in the whole collection, TF is the frequency of t in the collection, and $\#(token)$ is the number of tokens in the collection.

Using KL, the query term weight qtw is also determined by Equation (2.25), while the upper bound of $w(t)$ is given by:

$$\lim_{TF \to tf_x} w(t) = \frac{TF_{max} \cdot \log_2 \frac{\#(token)}{l_x}}{l_x} \tag{2.27}$$

where TF_{max} is the collection frequency TF of the term with the maximum $w(t)$ in the top-ranked documents, and l_x is the length of the feedback documents.

We use Bo1 term weighting model in Chapter 6 as it produces better retrieval performance than KL in the MAP evaluation measure.

2.4.2 Collection Enrichment

Cao et al. (2008) showed that the performance of query expansion is highly dependent on the quality of the terms added from the pseudo-relevant set, which suggests that the selected collection for building the pseudo-relevant feedback is an important factor for query expansion. Moreover, several studies have shown that, for some queries, it can be advantageous to obtain the pseudo-relevant feedback on a larger and higher-quality external collection (Diaz & Metzler, 2006; Kwok & Chan, 1998; Peng et al., 2009).

The process of expanding a query from an external collection and retrieving from the local collection using the expanded query is called *collection enrichment* (Kwok & Chan, 1998), or external query expansion (Diaz & Metzler, 2006). The collection enrichment algorithm is similar to the query expansion algorithm. The only difference is at the first step: query expansion builds pseudo-relevant feedback from the local collection while collection enrichment constructs a pseudo-relevant feedback from the external collection.

original query	radio astronomy
query expansion	astronomy astronomic cosmic frequency interfere radio signal spectrum telescope wave
collection enrichment	antennae array astronomy astronomic interferometric jansky radio ryle telescope wavelength

Table 2.2: An illustrative example of the query expansion and collection enrichment algorithms.

Table 2.2 shows an example of expanding a query by using query expansion and collection enrichment. The query "radio astronomy" is from the TREC 2007 Enterprise track topic set, the local collection is the CERC test collection (Bailey et al., 2007) and the external collection is the English Wikipedia corpus, which is a snapshot from August 2008 and contains over 3 million articles written collaboratively by users worldwide. We note that the query is short and only contains two terms. Moreover, many astronomy-related terms are added to enrich the initial short query by using either query expansion or collection enrichment, such as "spectrum", "jansky" and "telescope". In addition, there are some differences in the added terms between query expansion and collection enrichment. For example, the term "cosmic" only appears in the query that was obtained by using query expansion, while the term "jansky" is only included in the query that was created by using the collection enrichment algorithm. The collection enrichment technique will be used later in Chapter 6.

2.4.3 Query Performance Predictors

Many researchers have shown the overall effectiveness of the query expansion and collection enrichment techniques (Diaz & Metzler, 2006; Kwok & Chan, 1998; Rocchio, 1971*b*; Salton et al., 1983*b*). However, for some "difficult" queries, the retrieval performance obtained can be decreased after applying query expansion (Amati et al., 2004). Difficult queries are those where either the query terms cannot agree on the top results or most of the terms do agree except for a few outliers (Yom-Tov et al., 2005). Amati et al. (2004) showed that the retrieval performance can be boosted by avoiding the application of query expansion on

Predictor	**Type**	**Single**	**Co-occur**
AvICTF	Pre-retrieval	$\surd$	$\times$
AvIDF	Pre-retrieval	$\surd$	$\times$
$\gamma 1$	Pre-retrieval	$\surd$	$\times$
$\gamma 2$	Pre-retrieval	$\surd$	$\times$
AvPMI	Pre-retrieval	$\surd$	$\surd$
QS	Pre-retrieval	$\surd$	$\times$
CS	Post-retrieval	$\surd$	$\times$
WIG	Post-retrieval	$\surd$	$\surd$
QF	Post-retrieval	$\surd$	$\times$

Table 2.3: Overview of the techniques employed by the predictors. Columns *Single* and *Co-occur* refer to the use of the statistics of single query terms and the co-occurrence of multiple query terms, respectively.

the “difficult” queries.

Query performance predictor was proposed to predict whether a given query is an “easy” or a “difficult” query for a given document collection (Cronen-Townsend et al., 2002; Hauff et al., 2008). This kind of prediction mainly relies on the statistics of the collection for this given query, such as query term frequency in the collection and the number of documents containing the query term.

Many query performance predictors have been proposed in IR (Cronen-Townsend et al., 2002; Hauff et al., 2008; He & Ounis, 2006; Zhou & Croft, 2007) and are classified into two types: pre-retrieval predictors and post-retrieval predictors. Generally speaking, pre-retrieval predictors only rely on the statistics of the collection and the query while post-retrieval predictors are more reliant on the statistics of the top ranked documents for the query. In this section, several query performance predictors are presented, including 6 pre-retrieval predictors (Section 2.4.3.1) and 3 post-retrieval predictors (Section 2.4.3.2). An overview of the techniques used by these predictors can be found in Table 2.3.

2.4.3.1 Pre-retrieval Predictors

Average Inverse Document Frequency Inverse Document Frequency (IDF) is a measure of the general importance of a term, which has been integrated into many weighing models (e.g. BM25) for diminishing the weight of terms that occur frequently in the collection and increasing the weight of terms that occur rarely. It is computed as follows:

$$IDF(Q) = \sum_{i=1}^{|Q|} IDF(q_i) = \sum_{i=1}^{|Q|} \log \frac{N}{N_{q_i}} \tag{2.28}$$

where N is the total number of documents in the whole collection and N_{q_i} is the number of documents containing the query term q_i.

Based on the IDF measure, Hauff et al. (2008) proposed the Average Inverse Document Frequency (AvIDF) predictor, which is the mean of the inverse document frequency of the query terms, given as follows:

$$AvIDF(Q) = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} log \frac{N}{N_{q_i}} \tag{2.29}$$

where $|Q|$ is the number of terms in the given query Q, N is the total number of documents in the whole collection, and N_{q_i} is the number of documents containing the query term q_i.

Average Inverse Collection Term Frequency In (Kwok, 1996), Inverse Collection Term Frequency (ICTF), which is based on the frequency of query terms in the collection, was proposed to measure the quality of a query term:

$$ICTF(Q) = \log \prod_{i=1}^{|Q|} \frac{\#(token) - TF(q_i)}{TF(q_i)} \tag{2.30}$$

where q_i is a query term, $\#(token)$ is the total number of tokens in the collection and $TF(q_i)$ is the number of occurrences of the query term q_i in the collection.

The ICTF measure is similar to Inverse Document Frequency (IDF) except that it takes into account term frequency. Based on the idea of ICTF, He & Ounis (2006) proposed a query performance predictor, called Average Inverse Collection Term Frequency (AvICTF), given as follows:

$$AvICTF(Q) = \frac{log_2 \prod_{i=1}^{|Q|} \frac{\#(token)}{TF(q_i)}}{|Q|} \tag{2.31}$$

where $|Q|$ is the number of terms in the given query Q, $TF(q_i)$ is the number of occurrences of the query terms q_i in the whole collection, and $\#(token)$ is the total number of tokens in the collection.

$\gamma 1$ and $\gamma 2$ For each query term, an Inverse Document Frequency score can be calculated, which describes the informativeness that such query term carries. Pirkola & Järvelin (2001) observed that the difference between the resolution power of the query terms, as represented by the IDF scores, could affect the retrieval effectiveness of an IR system. Based on this observation, He & Ounis (2006) hypothesised that the distribution of the IDF scores can be used to predict the query performance and proposed two different query performance predictors, called $\gamma 1$ and $\gamma 2$. In particular, they employ INQUERY's IDF formula (Allan et al., 1995) to compute the IDF score for each query term, given as follows:

$$IDF(q_i) = \frac{log_2 \frac{N+0.5}{N_{q_i}}}{log_2(N+1)} \tag{2.32}$$

where, for each query term q_i from the given query Q, N is the total number of documents in the whole collection, and N_{q_i} is the number of documents in which the query term q_i appears.

$\gamma 1$ is defined as the standard deviation of the query terms' IDF scores, given as follows:

$$\gamma 1(Q) = \sigma_{IDF(q_i)} \tag{2.33}$$

$\gamma 2$ is defined as the maximum $IDF(q_i)$ score divided by the minimum $IDF(q_i)$ score, given as follows:

$$\gamma 2(Q) = \frac{\max\limits_{q_i} IDF(q_i)}{\min\limits_{q_i} IDF(q_i)} \tag{2.34}$$

where $\max_{q_i} IDF(q_i)$ is the maximum IDF scores among the query terms, and $\min_{q_i} IDF(q_i)$ is the minimum IDF scores among the query terms.

Averaged Pointwise Mutual Information The Averaged Pointwise Mutual Information (AvPMI) predictor was proposed to measure the mutual dependence of each pair of query terms in the collection (Hauff et al., 2008), given as follows:

$$AvPMI(Q) = \frac{1}{|(q_i, q_j)|} \sum_{(q_i,q_j)\in Q} log_2 \frac{P(q_i, q_j)}{P(q_i) \cdot P(q_j)} \tag{2.35}$$

where $|(q_i, q_j)|$ is the number of the possible pairs of all query terms, $P(q_i, q_j)$ is the probability that query terms q_i and q_j co-occur in a document, $P(q_i)$ and $P(q_j)$

are the probability that query terms q_i and q_j occur in a document individually. These probabilities are computed as follows:

$$P(q_i, q_j) = \frac{N_{q_i,q_j}}{N} \tag{2.36}$$

$$P(q_i) = \frac{N_{q_i}}{N} \tag{2.37}$$

$$P(q_j) = \frac{N_{q_j}}{N} \tag{2.38}$$

where N_{q_i,q_j} is the number of documents containing both query terms q_i and q_j, N_{q_i} is the number of documents containing query term q_i, N_{q_j} is the number of documents containing query term q_j, and N is the total number of documents in the whole collection.

Query Scope Plachouras et al. (2003) proposed the notion of Query Scope (QS), which is defined as a decreasing function of the size of documents that contain at least one query term. The estimation of query scope is based on the statistical evidence that is obtained from a set of retrieved documents. In (Plachouras et al., 2003), query scope has been used to decide whether or not to apply the combination of content and hyperlink analyses for a given query. Following (Plachouras et al., 2003), He & Ounis (2006) defined the query scope predictor as follows:

$$QS(Q) = -log\frac{N_{q'}}{N} \tag{2.39}$$

where, for a given Q, $N_{q'}$ is the number of documents containing at least one of the query term from Q, and N is the total number of documents in the collection.

Equation (2.39) shows that the QS score for a given query Q is a decreasing function of $N_{q'}$. A higher $N_{q'}$ score results in a lower QS score.

2.4.3.2 Post-retrieval Predictors

Clarity Score In (Cronen-Townsend et al., 2002), the concept of Clarity Score (CS) was introduced, which is used to measure the ambiguity of a query in relation to the collection of documents being searched. For this predictor, the query and the collection are represented by a query language model and a collection language model, respectively. In addition, the Kullback-Leibler divergence estimation method is employed to compute the degree of ambiguity, as follows:

$$CS(Q) = \sum_{w \in V} P(w|Q) log_2 \frac{P(w|Q)}{P(w|C)} \tag{2.40}$$

where w is a term included in the entire vocabulary V of the collection, $P(w|C)$ is a collection language model, and $P(w|Q)$ is a query language model.

The collection language model is computed as follows:

$$P(w|C) = \frac{TF(w)}{\#(token)} \tag{2.41}$$

where $TF(w)$ is the number of occurrences of term w in the whole collection, and $\#(token)$ is the total number of tokens in the whole collection.

The query language model is computed as follows:

$$P(w|Q) = \sum_{d \in D} P(w|d)P(d|Q) \tag{2.42}$$

where d is a document included in the retrieved document set D for a given Q, and $P(w|d)$ can be computed by using a document language model:

$$\begin{aligned} P(w|d) &\simeq \lambda \cdot P(w|d) + (1-\lambda) \cdot P(w) \\ &= \lambda \cdot \frac{tf_w}{l} + (1-\lambda) \cdot \frac{TF_w}{\#(token)} \end{aligned} \tag{2.43}$$

where tf_w is the number of occurrences of term w in the document d, TF_w is the number of occurrences of term w in the collection C, l is the total number of tokens in the document d, $\#(token)$ is the total number of tokens in the collection C, and λ is a parameter between 0 and 1.

After Bayesian inversion, $P(d|Q)$ is estimated as follows:

$$P(d|Q) = \frac{P(d) \cdot P(Q|d)}{P(Q)}$$

where $P(Q)$ can be ignored as it is the same for all documents, $P(d)$ is the prior probability of document d, and $P(Q|d)$ can be computed by using a document language model with a linear smoothing:

$$P(Q|d) = \prod_{w \in Q} P(w|d) \tag{2.44}$$

where $P(w|d)$ is computed as Equation (2.43).

Weighted Information Gain Most query performance predictors are based on the statistics of individual query terms, hence ignoring term dependency. In (Zhou & Croft, 2007), a query performance predictor, called Weighted Information Gain (WIG), was proposed to incorporate both single terms and phrase terms for predicting the performance of a given query. It is given as follows:

$$WIG(Q) = \sum_{d \in C} weight(Q,d) \log \frac{P(Q,d)}{P(Q,C)} \tag{2.45}$$

where

$$weight(Q,d) = \begin{cases} \frac{1}{K} & \text{if } d \in D \text{, where } D \text{ is a document set sampled from the top } K \text{ retrieved documents for the given } Q \\ 0 & \text{otherwise} \end{cases} \tag{2.46}$$

By using the MRF model (Section 2.5.1), $\log P(Q,d)$ is computed as follows:

$$\log P(Q,d) = -\log Z_1 + \sum_{f \in F} \lambda_f \log P(f|d) \tag{2.47}$$

where Z_1 is a constant that ensures the sum of $P(Q,d)$ is equal to 1. F contains both single query terms and phrase terms obtained by using the MRF model. For example, assuming that query Q is "query performance predictor", the single query terms contained in F are "query", "performance" and "predictor", and the phrase terms included in F are "query performance", "performance predictor" and "query predictor". The details about how to generate phrase terms can be found at Section 2.5. $P(f|d)$ can be computed by using a document language model linearly smoothed with the collection language model:

$$\begin{aligned} P(f|d) &\simeq \alpha \cdot P(f|d) + (1-\alpha) \cdot P(f) \\ &= \alpha \cdot \frac{tf_f}{l} + (1-\alpha) \cdot \frac{TF_f}{\#(token)} \end{aligned} \tag{2.48}$$

where tf_f is the number of occurrences of term f in the document d, TF_f is the number of occurrences of term f in the collection C, l and $\#(token)$ are the total number of tokens in the document d and collection C, respectively, and α is a parameter between 0 and 1. λ_f plays a dual role, which assigns different weights to the term and the phrase, given as follows:

$$\lambda_f = \begin{cases} \dfrac{\lambda}{\sqrt{\#(ST)}} & \text{if } f \in ST \\ \\ \dfrac{1-\lambda}{\sqrt{\#(PT)}} & \text{if } f \in PT \end{cases} \tag{2.49}$$

where λ is a parameter; $\#(ST)$ and $\#(PT)$ are the number of single terms and phrase terms, respectively.

Similar to $\log P(Q,d)$, $\log P(Q,C)$ can be written as:

$$\log P(Q,C) = -\log Z_2 + \sum_{f \in F} \lambda_f \log P(f|C) \tag{2.50}$$

where Z_2 is a constant and $P(f|C)$ can be computed as:

$$P(f|C) = \frac{TF(f)}{\#(token)} \tag{2.51}$$

where $TF(f)$ is the number of occurrences of term f in the whole collection and $\#(token)$ is the total number of tokens in the whole collection.

When constants Z_1 and Z_2 are dropped, Equation (2.45) can be reformulated as follows:

$$WIG(Q) = \frac{1}{K} \sum_{d \in D} \sum_{f \in F} \lambda_f \cdot log \frac{P(f|d)}{P(f|C)} \tag{2.52}$$

Query Feedback A ranked list of documents can be obtained by submitting a query to an IR system. Zhou & Croft (2007) viewed the retrieval system as a noisy channel and assumed that, by going through the channel, the query Q becomes corrupted and is transformed to a ranked list of documents. Based on this assumption, they consider the prediction of the retrieval effectiveness as the task of finding a way to measure the degree of corruption.

To measure the degree of corruption, a query performance predictor, called Query Feedback (QF), was proposed (Zhou & Croft, 2007), given as follows:

$$QF(Q) = \frac{\#(Co)}{K} \tag{2.53}$$

where, for a given query Q, a new query Q' can be generated by using a query expansion technique based on a list of the top K retrieved documents which are obtained by submitting Q to an IR system. With this new query Q', we can receive another list of top K retrieved documents after submitting Q' to this IR system. The number of documents occurring in both lists is denoted as $\#(Co)$. Hence, this predictor calculates the proportion of the documents co-occurring in both document lists.

2.5 Term Dependency

Most probabilistic document weighting models, such as BM25 and language modelling, assume that terms in both queries and documents are statistically independent and rank documents using the statistics of single query term in documents. However, this assumption does not always hold. For example, it is highly likely to see the term *retrieval* if the term *information* occurs in IR conference proceedings. This fact suggests that the occurrences of certain pairs of terms are correlated, which is also called term dependency.

A fair amount of research has been conducted to model term dependency (Fagan, 1987; Gao et al., 2004; Losee, 1994; Metzler & Croft, 2005; Nallapati & Allan, 2002; Peng et al., 2007; Srikanth & Srihari, 2002; van Rijsbergen, 1977). Fagan (1987) investigated term dependency through the use of non-syntactic (statistical) phrase indexing. In addition, five different features for identifying phrases have been proposed (shown in Table 2.4). The experimental results on a range of document collections suggest that no single feature of phrase identification could consistently enhance the retrieval performance. For some document collections, significant improvements were obtained when a phrase was defined as any two terms co-occuring in a query or a document with unlimited distance. However, for other document collections, the phrase identification method only makes a marginal or even has a negative effect on retrieval performance.

Losee (1994) proposed the use of the Bahadur Lazarsfeld Expansion (BLE) for modelling the term dependency between all terms instead of only two terms. However, the generalised term dependency model does not bring significant benefit in retrieval effectiveness, compared to the approach that is based on the term

Feature	Description
Length	The maximum number of elements in a phrase.
Domain	The place that the elements of a phrase co-occur, which could be a document, a query, or a sentence.
Proximity	Elements of a phrase must co-occur within a specified distance in the specified domain.
DFh	The document frequency threshold for the elements of a phrase. The document frequency of a element is defined as the number of documents in which the element occurs at least once.
DFp	The document frequency threshold for a phrase. The document frequency of a phrase is defined as the number of documents in which the phrase occurs at least once.

Table 2.4: The proposed features for phrase identification.

independency assumption. This is due to the complex nature of the model, which sometimes results in more loss from the estimation error than gain from modelling term dependency.

Gao et al. (2004) proposed to extend the language modelling for IR approach with a dependency structure, called *linkage*, which is inspired by link grammar (Lafferty et al., 1992; Pietra et al., 1994). In the proposed model, a linkage between a pair of terms in the scale of a sentence is created, by taking into account several linguistically motivated constraints (e.g. planar, acyclic). As illustrated in Figure 2.1, a linkage between “affirmative” and “action” is created under the linguistically motivated constraints. Their model has shown its effectiveness on a number of test collections. However, the model only considers the dependency between pairs of terms. In addition, it needs to build a linkage information for each query, which is time-consuming.

Other work uses syntactical and lexical phrase detection techniques for query refinement (Arampatzis et al., 2000; Mitra et al., 1997; Pickens & Croft, 2000). However, all techniques have been shown to yield little or no improvement in retrieval effectiveness. Mishne & de Rijke (2005) proposed a shallow approach, also called *everything-is-a-phrase*, for modelling term dependency. They explored the use of phrase and proximity terms in the context of Web retrieval. In their approach, a phrase term is defined as the subset of consecutive terms from a

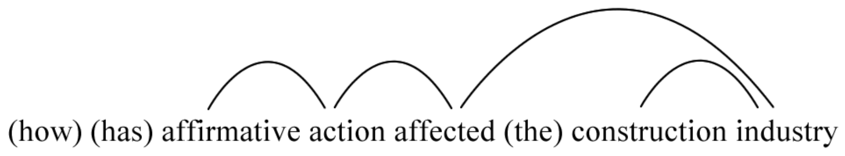

(how) (has) affirmative action affected (the) construction industry

Figure 2.1: An example of the linkage between terms in a sentence, where stop words are bracketed.

given query, which do not need to be actual phrases, either in the syntactical or statistical sense. For example, for the query "Ireland consular information sheet", three phrases are created if we set the phrase length to two:

"Ireland consular" "consular information" "information sheet"

Similar to phrase definition, a proximity term is defined as a subset of words from a given query within a specific distance. If we take the same aforementioned query as an example, five proximity terms are created if we set the maximum distance between the query terms to 3:

"Ireland consular" "Ireland information" "consular information"
"consular sheet" "information sheet"

Experiments on Web retrieval tasks have shown that the proposed phrase building approaches (Mishne & de Rijke, 2005) can bring some benefit to the retrieval effectiveness. In addition, a strong positive impact on the retrieval effectiveness has been observed when the proposed approaches are applied on short queries (i.e. 2 or 3 terms). The proposed approaches also bring other practical benefits, such as low computational overheads.

2.5.1 Term Dependency via Markov Random Fields

Metzler & Croft (2005) developed a general framework for modelling term dependency via Markov Random Fields (MRF) in a language modelling framework. In particular, three variants of the MRF model, namely full independence (FI), sequential dependence (SD), and full dependence (FD), have been proposed.

For the full independence variant, the query terms are assumed to be independent of one another. As shown in Figure 2.2, the query "Buchanan bus station"

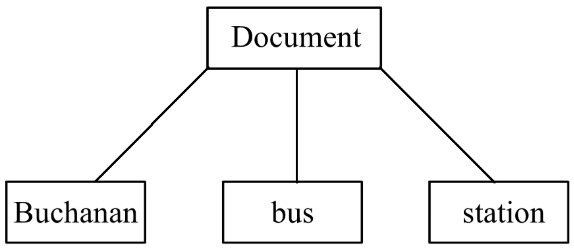


Figure 2.2: An example of the Markov random fields model with a full independence assumption.

is considered as three independent terms, namely "Buchanan", "bus", and "station". By using language modelling (Section 2.3.2) as the weighting scheme, the relevance score of each document to the query is computed as follows:

$$\begin{aligned} score_{FI}(d,Q) &= \sum_{i=1}^{|Q|} score(d,q_i) \\ &= \sum_{i=1}^{|Q|} log\left[(1-\alpha)\cdot\frac{tf_{q_i}}{l}+\alpha\cdot\frac{TF_{q_i}}{\#(token)}\right] \end{aligned} \tag{2.54}$$

where $score(d,q_i)$ is a language modelling estimate with the Jelinek-Mercer smoothing, $|Q|$ is the number of terms in the query Q, tf_{q_i} is the number of occurrences of query term q_i in the document d, and l is the total number of terms in the document d. TF_{q_i} is the number of occurrences of query term q_i in the collection and $\#(token)$ is the total number of tokens in the collection. α is the parameter of the Jelinek-Mercer smoothing, ranging from 0 to 1.

For the sequential dependence variant, only the neighbouring query terms are assumed to be dependent. Moreover, the terms contained in each phrase must be in the same order as in the query. As shown in Figure 2.3, the query "Buchanan bus station" is considered to be two phrases if we set each phrase to only contain two terms:

"Buchanan bus"   "bus station"

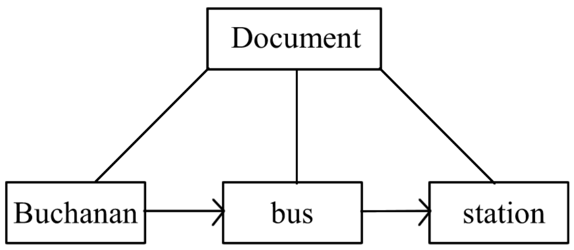


Figure 2.3: An example of the Markov random fields model with a sequential dependence assumption.

In this case, the relevance score of each document to the query is estimated as follows:

$$\begin{aligned} score_{SD}(d,Q) &= \sum_{i=1}^{|F_{SD}|} score(d, f_i) \\ &= \sum_{i=1}^{|F_{SD}|} log\left[(1-\alpha)\cdot\frac{tf_{f_i}}{l} + \alpha\cdot\frac{TF_{f_i}}{\#(token)}\right] \end{aligned} \tag{2.55}$$

where $|F_{SD}|$ denotes the number of phrases obtained under the sequential dependence assumption. tf_{f_i} and TF_{f_i} are the number of occurrences of phrase f_i in the document d and collection C, respectively.

The last variant is full dependence, Metzler & Croft (2005) assumed that the occurrence of non-neighbouring sets of query terms can also provide valuable evidence. Moreover, the order of the terms included in each phrase is relaxed. Under this assumption, as shown in Figure 2.4, the query "Buchanan bus station" is considered as six phrases if the length of each phrase is set to two:

"Buchanan bus"	"bus station"	"Buchanan station"
"bus Buchanan"	"station bus"	"station Buchanan"

Note that there is a free parameter, called *window size* (denoted as w_s), which is the maximum distance between the terms in the query. For example, the window

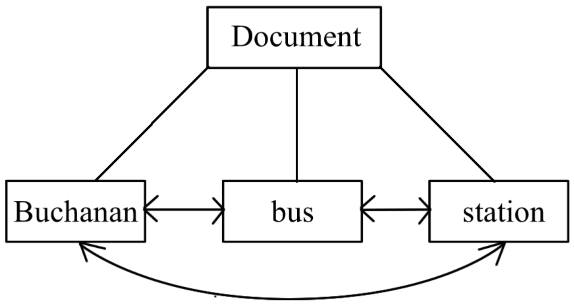


Figure 2.4: An example of the Markov random fields model with a full dependence assumption.

size between "Buchanan" and "station" in the given query is 3. In this case, the relevance score of each document to the query is estimated as follows:

$$score_{FD}(d,Q) = \sum_{i=1}^{|F_{FD}|} P(d,f_i) \tag{2.56}$$
$$= \sum_{i=1}^{|F_{FD}|} log\Big[(1-\alpha)\cdot\frac{tf_{f_i}}{l} + \alpha\cdot\frac{TF_{f_i}}{\#(token)}\Big]$$

where $|F_{FD}|$ denotes the number of phrases obtained under the full dependence assumption. Based on these different variants, the final relevance score of a document to the query is estimated by (Metzler & Croft, 2005):

$$score(d,Q) = \sum_{v} \lambda_v \cdot score_v(d,Q) \tag{2.57}$$
$$= \lambda_{FI}\cdot score_{FI}(d,Q) + \lambda_{SD}\cdot score_{SD}(d,Q) + \lambda_{FD}\cdot score_{FD}(d,Q)$$

where λ_{FI}, λ_{SD} and λ_{FD} are the parameter of each variant of the MRF model. In addition,

$$\lambda_{FI} + \lambda_{SD} + \lambda_{FD} = 1 \tag{2.58}$$

The variants of the MRF model cover many previously proposed retrieval and dependence models, such as unigram language modelling (Ponte & Croft, 1998), bigram language modelling (Song & Croft, 1999), and biterm language modelling (Srikanth & Srihari, 2002). Extensive experiments have shown the effectiveness of the MRF model on various search tasks (Metzler & Croft, 2005).

Metzler (2008) also discussed how to integrate the MRF term dependency model into the BM25 document weighting model, given as follows:

$$score(d,Q) = \sum_{p \in F_{SD}} \frac{(k_1+1)\cdot pf}{k_1 \cdot ((1-b)+b\cdot \frac{l}{avg\_l}) + pf} \cdot \log \frac{N-n+0.5}{n+0.5} \tag{2.59}$$

$$score(d,Q) = \sum_{p \in F_{FD}} \frac{(k_1+1)\cdot pf}{k_1 \cdot ((1-b)+b\cdot \frac{l}{avg\_l}) + pf} \cdot \log \frac{N-n+0.5}{n+0.5} \tag{2.60}$$

where F_{SD} and F_{FD} is the phrase set that is obtained under sequential dependence assumption and full dependence assumption, respectively. pf is the number of occurrences of the phrase p in d, N is the number of documents in the collection, n is the number of documents in which the phrase p appears, l is the document length, avg_l is the mean of the document length in the collection. k_1 and b are the parameters of the BM25 document weighting model.

2.5.2 Term Dependency in DFR

In (Peng et al., 2007), we proposed to incorporate term dependency into the DFR framework. This model assigns scores to each phrase created under a specific assumption, in addition to the single query terms. Hence, the score of a document d for a query Q is given as follows:

$$score(d,Q) = \lambda_1 \cdot \sum_{t \in Q} score(d,t) + \lambda_2 \cdot \sum_{p \in Q_2} score(d,p) \tag{2.61}$$

where $score(d,t)$ is the score assigned to a query term t in the document d, p corresponds to a phrase that appears within the query Q, $score(d,p)$ is the score assigned to phrase p in the document d, and Q_2 is a set of phrases, as defined below. The two scores are combined linearly using λ_1 and λ_2 as weights. In Equation (2.61), the score $\sum_{t \in Q} score(d,t)$ can be estimated by any DFR weighting model, such as PL2 (Equation (2.20)).

This model considers the same three possible variants as the MRF model for modelling the dependency between query terms:

- Full independence: the weighting model introduced only computes the first component of Equation (2.61), as it ignores the term dependencies between query terms ($\lambda_1 = 1$, $\lambda_2 = 0$).
- Sequential dependence: both components of Equation (2.61) ($\lambda_1 = 1$, $\lambda_2 = 1$) are computed, and in this case, Q_2 is the phrase set. In particular, each phrase is defined as the *ordered* neighbouring query terms.
- Full dependence: both components of Equation (2.61) ($\lambda_1 = 1$, $\lambda_2 = 1$) are computed, and in this case, Q_2 is the set of phrases, which are defined as the query terms without order limit.

The weight $score(d, p)$ of a phrase in a document is computed as follows:

$$score(d, p) = -\log_2(P_{p1}) \cdot (1 - P_{p2}) \tag{2.62}$$

where P_{p1} corresponds to the probability that there is a document in which a phrase p occurs a given number of times. P_{p1} can be computed with any Randomness model from the DFR framework (see Section 2.3.3.1). P_{p2} corresponds to the probability of seeing the phrase once more, after having seen it a given number of times. P_{p2} can be computed using any of the after-effect models in the DFR framework (see Section 2.3.3.2). The difference between $score(d, p)$ and $score(d, t)$ is that the former depends on counts of occurrences of the phrase p, while the latter depends on counts of occurrences of the query term t.

For example, for the DFR InL2 document weighting model (Amati, 2003):

$$score(d, Q) = \sum_{t \in Q} \frac{1}{tf + 1} \big(tf \cdot \log_2 \frac{N + 1}{n_t + 0.5}\big) \tag{2.63}$$

where n_t is the number of documents that the term t occurs in, N is the number of documents in the collection, tf is the number of occurrences of the term t in the document d. After applying term dependency, we obtain:

$$score(d, Q_2) = \sum_{p \in Q_2} \frac{1}{pf + 1} \big(pf \cdot \log_2 \frac{N + 1}{n_p + 0.5}\big) \tag{2.64}$$

where n_p corresponds to the number of documents in which the phrase p appears. pf is the number of occurrences of the phrase p in document d.

However, like term dependency in language modelling and BM25 (see Section 2.5.1), it is quite expensive to estimate the phrase frequency in the document collection. To avoid this, a different randomness model has been proposed in (Peng et al., 2007), which is based on the binomial randomness model, given as follows:

$$\begin{aligned} score(d,p) = \frac{1}{pfn+1} \cdot \Big(& - \log_2 (l - w\_s + 1)! + \log_2 pfn! \\ & + \log_2 (l - w\_s + 1 - pfn)! \\ & - pfn \log_2 (p_p) \\ & - (l - w\_s + 1 - pfn) \log_2 (p'_p) \Big) \end{aligned} \tag{2.65}$$

where l is the length of the document in tokens, $p_p = \frac{1}{l - w\_s + 1}$, $p'_p = 1 - p_p$, and pfn is the normalised frequency of the phrase p using Normalisation 2 (see Section 2.3.3.3):

$$pfn = pf \cdot \log_2 (1 + c_p \cdot \frac{avg\_l - w\_s + 1}{l - w\_s + 1})(c_p > 0) \tag{2.66}$$

pf is the frequency of the phrase p that appears within w_s tokens in the document, avg_l is the average document length in the collection, and c_p is a hyper-parameter that controls the normalisation applied to the phrase frequency against document length. The term dependency model in DFR will be used in Chapter 6 with the aim of building a strong baseline.

Experiments on several different test collections showed that both term dependency via MRF and term dependency in DFR can significantly improve retrieval effectiveness over a baseline that assumes the terms are independent (Metzler & Croft, 2005; Peng et al., 2007). In particular, the modelling of full dependency can significantly outperform sequential dependency.

2.6 Query Independent Features

The techniques introduced above for ranking documents are dependent on the statistics of the terms included in a given query. There is a kind of feature, called a query independent feature, which is independent of the statistics of a given query. A query independent feature, relating perhaps to linkage or usage, can be transformed into a static, per-document relevant weight for use in ranking (Craswell et al., 2005). Several studies have shown that the application of query independent features, such as PageRank (Brin & Page, 1998) and URL depth (Kamps et al., 2004), can enhance the retrieval effectiveness of a Web IR system (Cai et al., 2004; Craswell et al., 2005; Kraaij et al., 2002; Metzler et al., 2005). In this work, we classify them into three categories: link-based features, URL-based features and other features.

2.6.1 Link-based Query Independent Features

Indegree Documents in the Web are connected through hyperlinks. A hyperlink is a connection between a source and a target document. There is a simple assumption that a hyperlink from document A to document B stipulates that the document A's author considers document B to be valuable. A high number of incoming links often indicates that many documents' authors consider the given document to be of a high quality. This feature is called *indegree* (InD), which counts the number of incoming links for each document, given as follows:

$$score_{InD}(d) = \#(inlink, d) \tag{2.67}$$

where $\#(inlink, d)$ is the number of incoming link for document d. As an illustration, in Figure 2.5, documents have different number of incoming links:

$$\begin{array}{lll} score_{InD}(A) = 4 & score_{InD}(B) = 3 & score_{InD}(C) = 2 \\ score_{InD}(D) = 2 & score_{InD}(E) = 1 & \end{array}$$

Outdegree In contrary to indegree, there is another query independent feature called *outdegree* (OutD), which counts the number of outgoing links from a document, given as follow:

$$score_{OutD}(d) = \#(outlink, d) \tag{2.68}$$

where $\#(outlink, d)$ is the number of outgoing links from document d. For Figure 2.5, we obtain:

$$\begin{array}{lll} score_{OutD}(A) = 2 & score_{OutD}(B) = 1 & score_{OutD}(C) = 2 \\ score_{OutD}(D) = 1 & score_{OutD}(E) = 1 & \end{array}$$

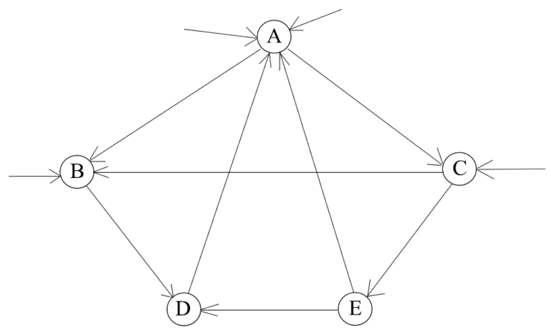


Figure 2.5: An illustration of the link structure of documents.

PageRank Brin & Page (1998) proposed *PageRank* (PR), which not only counts the number of incoming links to a document, but also takes the quality of these links into account. PageRank is based on citation analysis which is used to examine the frequency, patterns and graphs of citations in articles and books (Harris, 2006). PageRank is a probability distribution, which is employed to represent the likelihood that a document will be selected if a user makes a random click.

The PageRank feature score of a given document is computed as follows:

$$score_{PR}(d) = \frac{(1-\lambda)}{N} + \lambda \cdot \sum_{d_i \in L(d)} \frac{score_{PR}(d_i)}{score_{OutD}(d_i)} \tag{2.69}$$

where $L(d)$ is a document set, which contains all the documents with a link to document d, $score_{OutD}(d_i)$ is the number of outgoing links from document d_i, N is the number of documents in the collection. λ_{PR} is a damping factor and usually set to $\lambda_{PR} = 0.85$ (Brin & Page, 1998). Note that the computation of the PageRank feature score is an iteration process, which stops until a convergence is met, e.g. most documents' PageRank scores do not change.

2.6.2 URL-based Query Independent Features

URL Type A Uniform Resource Locator (URL), which contains a string of symbols, defines the unique location of a document on the Web. Kraaij et al. (2002) proposed the use of *URL type* (UT) for identifying homepages. This is motivated by the fact that homepage documents only contain a domain name, optionally followed by the "index.html" string. In particular, they classify a URL string into four different categories:

- **root**: a document URL contains either only a domain name or a domain name followed by "index.html" (e.g. `http://www.sigir.org`).
- **subroot**: a document URL contains a domain name followed by a single directory, optionally followed by "index.html" (e.g. `http://www.sigir.org/sigirlist/`).
- **path**: a document URL contains a domain name followed by a single directory or many directories, but not ending in a file name other than "index.html" (e.g. `http://www.sigir.org/sigirlist/issues/`).
- **file**: a document URL ends in a filename other than "index.html" (e.g. `http://www.sigir.org/resources.html`).

For a document, its URL type score is computed as follows:

$$score_{UT}(d) = \frac{\#(t_i, D)}{\#(t_i, C)} \tag{2.70}$$

where $\#(t_i, D)$ and $\#(t_i, C)$ are the number of documents that have the same URL type t_i with document d in the query relevance assessment and document collection, respectively. Therefore, the estimation of the URL type score for each document is based on a training dataset.

URL Depth The URL string can be divided into many components by the symbol '/', excluding the "http://" component. The *URL depth* (UD) feature counts the number of components after the division. For example, the URL `http://www.firstgov.gov/topics/science.html` can be divided into 3 components:

$$www.firstgov.gov \quad topics \quad science.html$$

The URL depth feature score for a given document is defined as follows:

$$score_{UD}(d) = \#(component) \tag{2.71}$$

where $\#(component)$ is the number of components after the division.

URL is a string of symbols, the *URL length* (UL) feature simply counts the number of symbols of a document URL, given as follows:

$$score_{UL}(d) = \#(symbol) \tag{2.72}$$

where $\#(symbol)$ is the number of symbols in the URL of document d, excluding the "http://" component. For example, for the URLs given below:

```
http://www.sigir.org
http://www.sigir.org/sigirlist/
http://www.sigir.org/sigirlist/issues/
http://www.sigir.org/resources.html
```

their UL feature scores are equal to 13, 24, 31 and 28, respectively.

2.6.3 Other Query Independent Features

Apart from link-based and URL-based query independent features, there are several other features, such as information to noise ratio and click distance.

Information to Noise Ratio Zhu & Gauch (2000) proposed a feature to measure the quality of a document, called *information to noise ratio* (ITNR), given as follows:

$$score_{ITNR}(d) = \frac{\#(token_{after})}{\#(token_{before})} \tag{2.73}$$

where $\#(token_{after})$ is the number of tokens contained in the document d after preprocessing, such as removing stop words and HTML tags; $\#(token_{before})$ is the raw size of the document d, i.e., the number of tokens contained in the document.

Click Distance Click distance (CD) is a link metric which measures the minimum number of clicks it takes to reach a Web document from a given root (Craswell et al., 2005):

$$score_{CD}(d) = \#(click) \tag{2.74}$$

where $\#(click)$ is the minimum number of clicks from the root to document d.

As shown in Figure 2.6, document A is the root, and there are several different ways to reach document J, one such path is $A \rightarrow C \rightarrow F \rightarrow J$, which takes 3

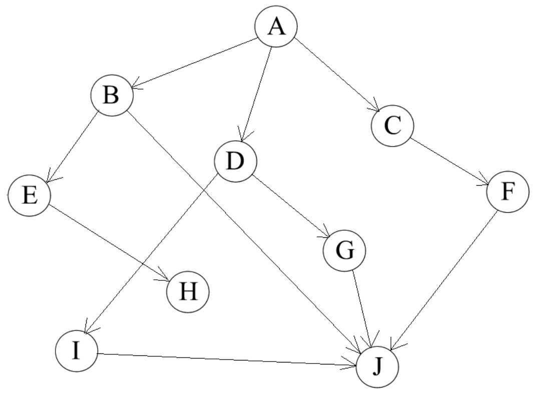


Figure 2.6: An illustration of the click distance feature.

clicks. However, the shortest path from A to J is $A \rightarrow B \rightarrow J$, which only takes 2 clicks. In this example, the click distance score for document J is:

$$score_{CD}(J) = |A \rightarrow B \rightarrow J| = 2$$

In addition, the click distance scores of other documents from the same root A are given as follows:

2.6.4 The Integration of Query Independent Feature

Several different Query Independent (QI) features have been introduced in the previous sections. In this section, we present how to integrate these QI features into a document weighting scheme. The language modelling approach (Section 2.3.2) automatically takes into account the QI features and treats them as a prior probability:

$$P(d|Q) = \frac{P(Q|d) \cdot P(d)}{P(Q)} \tag{2.75}$$

where $P(d)$ is the prior probability of the relevance of the document d for a given QI feature, such as PageRank.

In contrast to the language modelling approach, Craswell et al. (2005) treated the QI features as static, per-document relevance weights for use in ranking. In particular, they proposed the *FLOE* method for transforming a query-independent feature value into a relevance score. The method allocates a query-independent feature score for each document d as follows:

$$score(d, Q) = score_{QD}(d, Q) + score_{QI}(d) \tag{2.76}$$

where $score_{QD}(d, Q)$ is the query-dependent relevance score of d given a query Q and can be estimated by a document weighting scheme, such as BM25 (Equation (2.6)); $score_{QI}(d)$ is the query-independent relevance score for a given document d, estimated by $FLOE$ using a query-independent feature. $score(d, Q)$ is the final relevance score of document d given the query Q.

Craswell et al. (2005) proposed two different versions of the $FLOE$ method, we denote them as $FLOE^+$ and $FLOE^-$. The two versions of $FLOE$ are defined as follows:

$$FLOE^+(S, w, k, a) = w \cdot \frac{S^a}{k^a + S^a} \tag{2.77}$$

$$FLOE^-(S, w, k, a) = w \cdot \frac{k^a}{k^a + S^a} \tag{2.78}$$

where S is the query-independent feature score, w, k and a are parameters. With the same w, k, and a settings, in Equation (2.77), a document with a higher query-independent feature score attains a higher relevance score after the transformation, while in Equation (2.78), a document with a higher query-independent feature score attains a lower relevance score after the transformation. For example, when PageRank scores are mapped using $FLOE^+$, a document that has a high PageRank score is usually considered to be a high-quality document. On the other hand, when URL depth scores are transformed using $FLOE^-$, documents with shorter URL depth are usually seen as more authoritative than pages with longer URL depth.

2.7 Evaluation

In this chapter, we have presented many different retrieval techniques for building IR systems. A natural question arises on how to evaluate the performance of an IR system. The evaluation of an IR system is the process of assessing how well the system meets the information need of its users (Voorhees, 2001). This evaluation process normally consists of three components: a document collection, a set of information needs, which are expressed as queries, and a set of relevance assessments, which specify which documents are relevant to a specific query.

By submitting a query to an IR system, a set of documents in the collection is returned. With the relevance assessments set for this query, the performance of the IR system can be evaluated by examining whether each returned document is relevant to the query. The most commonly used evaluation measures for an IR system are *precision* and *recall*. Precision measures the percentage of the retrieved relevant documents in the retrieved set for a particular query, and recall measures the percentage of the retrieved relevant documents in the whole relevant set for a particular query.

Over the past decades, several standard test collections have been created with the aim of providing a consistent test bed and benchmark for evaluating the performance of IR system, such as the Cranfield collection (Cleverdon, 1962) which was a pioneer in conducting IR evaluation in a quantitative manner. In the Cranfield experiments, the small size of the test collection (1,398 abstracts) allowed the complete assessment of each document for all queries. However, as small test collections do not reflect the main issues in modern IR environments such as link analysis, larger collections were created, such as .GOV2 (25 million documents) (Clarke et al., 2004). With these large test collections, complete assessment of all documents for each query became impractical.

In the context of the Text REtrieval Conference (TREC), the generation of relevance assessments is based on the *pooling* technique (Harman, 1993), which is developed based on an idea from Spärck Jones & van Rijsbergen (1976). For each query, the top K returned documents (normally $K = 100$) from a set of participating IR systems are merged into a document pool (Voorhees & Harman, 2000). The relevance assessments are then conducted on the merged document pool, instead of all the documents in the test collection. As the document pool is built from different IR systems, the generated relevance assessments are not biased towards any particular IR system. However, when the size of the test collection increases, the recall value is overestimated as it is highly likely that many relevant documents have not been contained in the pool (Blair, 2002; Zobel, 1998). In particular, Zobel (1998) estimated that the document pool comprises a maximum of 50-70% of the total number of relevant documents.

Choosing the evaluation measures in TREC is task-oriented. Apart from precision and recall, the *average precision* measure rewards returning more relevant documents earlier. It is computed as the average of the precision values after each relevant document is retrieved (Manning et al., 2008). For example, if an IR system retrieves three relevant documents for a given query, at ranks 1, 3 and 7, the average precision of the system for this query is computed as $\frac{1}{3}(\frac{1}{1}+\frac{2}{3}+\frac{3}{7}) = 0.6984$. When there is only one relevant document for each query, the average precision measure is equivalent to the reciprocal rank (RR) of the first retrieved relevant document (Voorhees, 2008). The comparison of systems over a set of queries is performed by employing the mean of the above described evaluation measures, leading to mean average precision (MAP), mean reciprocal rank of the first retrieved relevant document (MRR).

While average precision measures the retrieval performance based on the full list of retrieved documents for each query, *R-precision* calculates the precision after R relevant documents have been returned (Manning et al., 2008), where R is the number of relevant documents in the collection for a particular query. Another measure is *precision at k* (denoted $P@k$), which calculates the proportion of relevant documents in the top k retrieved documents.

Buckley & Voorhees (2004) showed that current evaluation measures are not reliable on substantially incomplete relevance assessments. To overcome this issue, they proposed the binary preference (bpref) evaluation measure. This calculates a preference relation which measures whether documents judged relevant are returned ahead of those judged not relevant. There are some other evaluation measures which have been proposed in recent years, such as normalised Discounted Cumulative Gain (nDCG) (Järvelin & Kekäläinen, 2002) and inferred Average Precision (infAP) (Yilmaz & Aslam, 2006), which can also be used when the relevance judgements are incomplete. For the latest Web diversity task (Clarke et al., 2009), there are two official evaluation measures: $\alpha-$normalised discounted cumulative gain ($\alpha-$NDCG (Clarke et al., 2008)) and intent-aware precision (IA-P (Agrawal et al., 2009)). The $\alpha-$NDCG measure balances relevance and diversity by varying the value of α. The larger the α is, the more diversity is rewarded. In the case that $\alpha = 0$, the $\alpha-$NDCG measure is equivalent to the traditional NDCG measure. The IA-P measure extends the traditional precision measure by taking into account the possible aspects underlying a given initial query, as well as their relative importance.

2.8 Summary

We have presented an overview of various IR techniques in this chapter, from the indexing process to ranking and the evaluation of an IR system. In particular, several different document features for ranking information items have been described, such as document weighting models, query expansion, term dependency, and query independent document features. In the next chapter, we show how to build a ranking function by using these document features (Section 3.2). In addition, several state-of-the-art selective retrieval approaches are also presented (from Section 3.3 to Section 3.6).

Chapter 3

Learning to Rank and Selective Retrieval Approaches

3.1 Introduction

The previous chapter presented various retrieval techniques, including document weighting models, query expansion and term dependency. Each retrieval technique that was introduced can be treated as a document feature for ranking, e.g. the document relevance score estimated by the BM25 (Equation (2.6)) weighting model has been widely used as a document feature in IR (Craswell et al., 2005; Liu et al., 2007). In order to obtain a better retrieval performance, instead of using a single or a few features, there is a growing trend to learn appropriate weights for a large set of features (Cao et al., 2007; Geng et al., 2008; Taylor et al., 2008; Xu & Li, 2007). Such a set of weighted features for document ranking forms a ranking function, which is obtained by the use of a learning to rank technique. Several different learning to rank techniques are introduced in this chapter (Section 3.2).

However, most current work simply applies a learned ranking function to a given set of queries (Cao et al., 2007; Metzler, 2007; Taylor et al., 2008; Xu & Li, 2007), which ignores the fact that different ranking functions favour different queries. Various selective retrieval approaches have previously been proposed in IR (Geng et al., 2008; Peng et al., 2009; Plachouras, 2006; Plachouras & Ounis, 2004; Song et al., 2004; Yang et al., 2004). In this chapter, we compare a selection of retrieval approaches, including query dependent ranking (Section 3.3), selective collection enrichment (Section 3.4), selective Web IR (Section 3.5) and query type prediction (Section 3.6).

3.2 Learning to Rank

Learning to rank is to learn a ranking function by assigning a weight to each document feature, then using this obtained ranking function to estimate relevance scores for each document, and finally ranking these documents based on the estimated relevance scores (Liu et al., 2007; Xu & Li, 2007). In recent years, many effective learning to rank techniques have been proposed to build ranking functions, such as AdaRank (Xu & Li, 2007) and Ranking SVM (Herbrich et al., 2000; Joachims, 2002). These techniques can be classified into three groups based on their underlying approach to the problem (Liu, 2009): the pointwise approach, the pairwise approach and the listwise approach.

3.2.1 The Pointwise Approach

The most straightforward way to employ a machine learning technique for ranking documents is to use existing learning methods. This kind of strategy is in line with the pointwise approach, which estimates the exact relevance degree of each document for a given query. In general, the pointwise approach can be divided into two categories: regression-based and classification-based.

3.2.1.1 The Regression Based Technique

Regression is a statistical technique for estimating the relationship between a dependent variable and one or more independent variables (Hastie et al., 2001). Regression can be used as a descriptive method of data analysis (such as curve fitting) without relying on any assumptions about underlying processes generating the data (Cook & Weisberg, 1982). By using a regression-based technique, the output space contains a real-valued relevance score for each document. Fuhr (1989) proposed the use of least square polynomials for building a retrieval function. A class of polynomial retrieval functions is defined based on some heuristic assumptions, e.g., the loss expectations can be approximated by average values, and the function that best fits the available data is selected.

With a given query q and a set of retrieved documents $D = \{d_j\}_{j=1}^m$, the ground truth label for d_j is defined as a vector $\overrightarrow{y_j}$. For a binary judgement, $\overrightarrow{y_j} = (1, 0)$ and $\overrightarrow{y_j} = (0, 1)$ if the document d_j is judged as relevant or irrelevant, respectively. For a multiply ordered category, only the k-th element of the vector $\overrightarrow{y_j}$ is set to 1 and the remaining elements are set to 0, if the document d_j is judged as belonging to the k-th category.

Assuming an n-category judgement case, the retrieval function is defined as:

$$\overrightarrow{f(d_j)} = (f_1(d_j), f_2(d_j), ..., f_n(d_j))$$

where each element $f_k(d_j)$ is calculated by using the polynomial function, given as follows:

$$\begin{aligned} f_k(d_j) = w_{k,0} + w_{k,1} \cdot d_{j,1} + \cdots + w_{k,T} \cdot d_{j,T} \\ +w_{k,T+1} \cdot d_{j,1}^2 + w_{k,T+2} \cdot d_{j,1} \cdot d_{j,2} + \cdots \end{aligned} \tag{3.1}$$

where $d_{j,l}$ is the l-th feature in the feature vector of d_j, $w_{k,l}$ is the weight coefficient of $d_{j,l}$, and T is the number of document features.

In addition, the loss function is defined as the following square loss:

$$L(\overrightarrow{f(d_j)} : d_j, \overrightarrow{y_j}) = \|\overrightarrow{y_j} - \overrightarrow{f(d_j)}\|^2 \tag{3.2}$$

3.2.1.2 The Classification-Based Technique

To avoid the problem of treating the relevance label as a quantitative value, the classification-based technique has been proposed for building a retrieval function. Robertson & Spärck Jones (1988) firstly viewed IR as a classification problem and considered retrieval as a process of classifying the entire collection of documents into two categories: relevant and non-relevant. In the recent past, Nallapati (2004) explored the applicability of discriminative models, which are a class of models used in machine learning for modelling the dependence of an unobserved variable on an observed variable (Duda et al., 2001), for IR. In particular, two representative classification models, namely Maximum Entropy (ME) (Berger et al., 1996) and Support Vector Machines (SVM) (Burges, 1998), were studied.

The principle of ME is to model all that is known and assume nothing about the rest. The parametric form of the ME model can be expressed as follows (Nallapati, 2004):

$$P(R|d,Q) = \frac{1}{Z(Q,d)} exp(\sum_{i=1}^{n} \lambda_i f_i(d,Q)) \tag{3.3}$$

where $Z(Q,d)$ is a normalisation constant, n is the number of features, λ_i is the weight of the feature function $f_i(d,Q)$. The feature weights can be learned based on the training data using a fast gradient descent algorithm (Malouf, 2002).

The SVM model aims to separate two categories of training examples with the largest margin in a hyper-plane (Burges, 1998). The hyper-plane is a high dimension space, which is mapped from the feature space. It is expected that the larger the margin is, the better the generalisation of the classifier is. The discriminative model by using SVM is given by:

$$P(R|d,Q) = \overrightarrow{w} \bullet \phi(\overrightarrow{f(d,Q)}) + b \tag{3.4}$$

where $\overrightarrow{f(d,Q)}$ is the vector of features, $\overrightarrow{w}$ is the weight vector in kernel space, $\bullet$ denotes the inner product, b is a constant and ϕ is the mapping from the input space to the kernel space.

An example of SVM is shown in Figure 3.1. In the figure, the green line (H3) does not separate the 2 classes while both the blue line (H1) and the red line (H2) can separate the 2 classes. In particular, the red line performs better than the blue line as it produces a bigger margin between the 2 classes.

3.2.1.3 Summary

Both the regression based and the classification based techniques have been introduced in this section. There are still other pointwise approaches which have been previously proposed for ranking documents (Cooper et al., 1992; Cossock & Zhang, 2006; Gey, 1994). However, there are two main common issues with these pointwise techniques: First, as the loss function is estimated for each query, the overall loss will be dominated by the queries that have a large number of retrieved documents. For example, if q_1 has 1000 retrieved documents and q_2 has only 10 retrieved documents, then the loss function will be biased towards q_1 even though

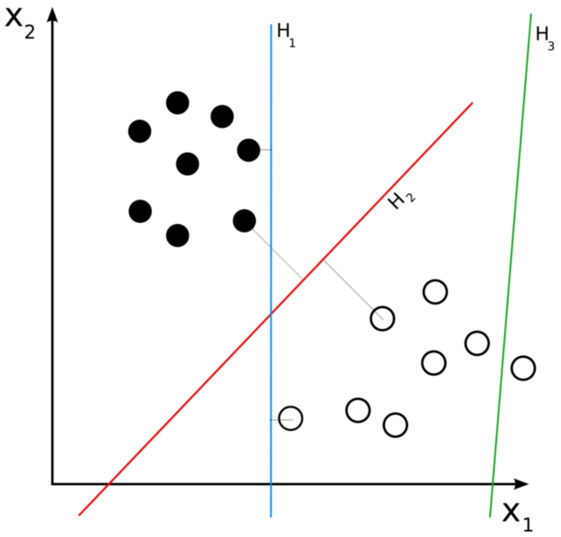


Figure 3.1: An illustration of the SVM classifier.

they should be treated equally. Second, as the pointwise approaches treat documents separately, the relative order between documents is invisible to the loss functions. This would result in the loss functions mainly focusing on irrelevant documents as a big proportion of a set of retrieved documents is irrelevant.

3.2.2 The Pairwise Approach

While the pointwise approach focuses on estimating the exact relevance degree of each document to a given query, the pairwise approach aims to rank a list of documents by investigating the relative order of pairs of documents. In an extreme case, for a list of documents, if all the document pairs are correctly ranked, then the full list of documents will be correctly ranked. For example, assuming that a document list contains three documents: d_1, d_2 and d_3; and that we obtain the following relative position between each pair of documents: $d_1 > d_2$, $d_1 > d_3$ and $d_2 < d_3$; then, these documents can be ranked as: $d_1 > d_3 > d_2$.

Several pairwise techniques have been previously proposed for ranking documents (Burges et al., 2005; Cao et al., 2006; Cohen et al., 1998; Freund et al., 2003; Herbrich et al., 2000; Joachims, 2002; Qin et al., 2007; Tsai et al., 2007). In this section, we introduce three representative techniques, namely RankNet (Burges et al., 2005), RankBoost (Freund et al., 2003) and Ranking SVM (Herbrich et al., 2000; Joachims, 2002).

3.2.2.1 RankNet

Burges et al. (2005) proposed the RankNet algorithm, which learns a retrieval function by employing a probabilistic cost function on a set of pairs of training examples. In particular, the cross entropy cost function is used in RankNet, given as follows:

$$L(f; d_u, d_v, y_{u,v}) = -\overline{P_{u,v}} \cdot logP_{u,v}(f) - (1 - \overline{P_{u,v}}) \cdot log(1 - P_{u,v}(f)) \tag{3.5}$$

where d_u and d_v are two documents, which are from the retrieved document set of a given query, $y_{u,v}$ is obtained based on the ground truth labels (for example, if the ground truth label indicates that document d_u should be ranked before document d_v, then $y_{u,v} = 1$), the target probability $\overline{P_{u,v}}$ is constructed according to $y_{u,v}$ (for example, $\overline{P_{u,v}}$ can be defined as $\overline{P_{u,v}} = 1$ if $y_{u,v} = 1$ and $\overline{P_{u,v}} = 0$ otherwise), and The modelled probability $P_{u,v}(f)$ is computed by using a logistic function, given as follows:

$$P_{u,v}(f) = \frac{exp(f(d_u) - f(d_v))}{1 + exp(f(d_u) - f(d_v))} \tag{3.6}$$

where $f(d_u)$ and $f(d_v)$ are the relevance score of d_u and d_v given by the retrieval function f.

In RankNet, a neural network is used to model and the gradient descent algorithm is employed as the optimisation algorithm to learn the retrieval function f. The effective performance of RankNet is observed on a real-world ranking problem with large amounts of data that is sampled from commercial search engines (Burges et al., 2005). In addition, RankNet was the first learning to rank algorithm used by commercial search engines (Liu, 2009).

3.2.2.2 RankBoost

Boosting refers to a general method of building a single strong learner by repeatedly constructing a weak learner with respect to a specific distribution and adding it to the strong learner (Freund & Schapire, 1999). A weak learner is defined to be a classifier which is only slightly correlated with the true classification. In contrast, a strong learner is a classifier that is arbitrarily well correlated with the true classification (Freund, 1990).

Based on the boosting technique, Freund et al. (2003) proposed an efficient learning technique, called RankBoost. Similar to other boosting algorithms, the RankBoost algorithm builds a document ranking function by combining several "weak" rankers of a set of document pairs. The learning algorithm for RankBoost is given in Algorithm 1:

```
Algorithm: RankBoost
Input: document pairs
Given: initial distribution $Dis_1$ on input document pairs
For $t = 1, ..., T$:
    • Train weak ranker $f_t$ based on distribution $Dis_t$.
    • Choose $\alpha_t$
    • Update $Dis_{t+1}(d_u, d_v) = \frac{Dis_t(d_u,d_v) \cdot exp(\alpha_t(f_t(d_u) - f_t(d_v)))}{Z_t}$
    where $Z_t$ is a normalisation factor (chosen so that
    $Dis_{t+1}$ will be a probability distribution).
Output: $f(d) = \sum_{t=1}^{T} \alpha_t f_t(d)$
```

Algorithm 1: The algorithm of RankBoost

where Dis_t is the distribution on document pairs, f_t is the weak ranker selected at the t-th iteration, and the coefficient α_t is the weight for linearly combining the weak ranker f_t. In particular, Freund et al. (2003) introduced three different methods for computing α_t:

1. First and most generally, for any given weak ranker f_t, it can be shown that Z_t, viewed as a function of α_t, has a unique minimum which can be found numerically via a simple binary search.

2. The second method is applicable in the special case that weak rankers take a value from $\{0, 1\}$. In this case, we can minimise Z_t analytically as follows: For $b \in \{-1, 0, +1\}$, let

$$W_{t,b} = \sum_{u,v:y_{u,v}=1} Dis_t(d_u, d_v) I_{\{f_t(d_u) - f_t(d_v) = b\}} \tag{3.7}$$

then

$$\alpha_t = \frac{1}{2} log \frac{W_{t,-1}}{W_{t,1}} \tag{3.8}$$

3. The third method is based on an approximation of Z_t, which is applicable when the weak ranker f_t takes a real value from $[0, 1]$. In this case, if we define:

$$r_t = \sum_{u,v:y_{u,v}=1} Dis_t(d_u, d_v)(f_t(d_u) - f_t(d_v)) \tag{3.9}$$

then

$$\alpha_t = \frac{1}{2} log \frac{1 + r_t}{1 - r_t} \tag{3.10}$$

Freund et al. (2003) tested the effectiveness of RankBoost on two different tasks: the first one is the meta-searching task, which attempts to combine the rankings of several Web search strategies; the second one is the movie-recommendation task, which is to rank movies for a user based on the movie rankings that are provided by other users. Experimental results showed that RankBoost performed just as well as the best retrieval strategy, whereas it consistently outperformed a regression approach (Hill et al., 1995) and the vector similarity approach (Breese et al., 1998) for the movie-recommendation task.

3.2.2.3 Ranking SVM

Support Vector Machine (SVM) has been widely and effectively used for binary classification in many fields. For instance, in information retrieval (IR), SVM is used to classify documents (Nallapati, 2004), or to estimate whether the most frequent terms in the pseudo-feedback documents are useful or not for query expansion (Cao et al., 2008). However, SVM cannot indicate the ranking sequence among multiple objects (e.g. documents) because it is a binary classifier. In contrast, in Web IR, generating a document ranking is the central research question.

In order to solve this issue, Ranking SVM (Herbrich et al., 2000; Joachims, 2002) has been proposed, which is a pairwise approach and based on the theory of SVM. During the training process for building a ranking function, it formalises a single ranking among k objects into many $(\binom{2}{k})$ rankings, where each ranking is between only two objects, which can be managed by using SVM. For example,

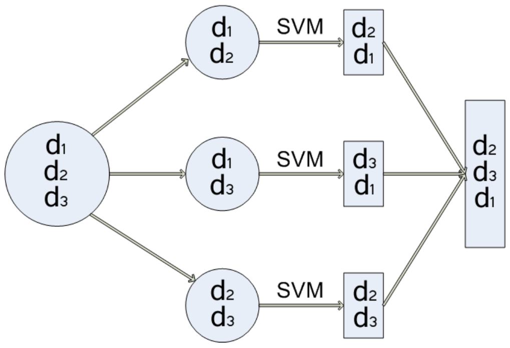


Figure 3.2: An illustration of the procedure of the Ranking SVM approach during the training process.

as shown in Figure 3.2, for a list of 3 documents: d_1, d_2, and d_3, instead of directly producing a one-time ranking for these documents (which is not possible with SVM), the Ranking SVM algorithm conducts $\binom{2}{3}$ ranking estimations. Each estimation is conducted between any two of these documents (namely $\{d_1, d_2\}$, $\{d_1, d_3\}$, and $\{d_2, d_3\}$) using SVM, and a final ranking is produced based on the three ranking estimations.

The mathematical formulation of Ranking SVM is given as follows:

$$\begin{gathered} minimise: \frac{1}{2}\overrightarrow{w} \cdot \overrightarrow{w} + C \sum_{u,v:y_{u,v}=1} \xi_{u,v} \\ subject\ to: \overrightarrow{w}(d_u - d_v) \geq 1 - \xi_{u,v}, if\ y_{u,v} = 1, \xi_{u,v} \geq 0 \end{gathered} \tag{3.11}$$

where $\overrightarrow{w}$ is a weight vector that is adjusted during the training process, $\xi_{u,v}$ is a variable that is introduced for the approximation purpose, C is a parameter that allows trading-off margin size against training error.

Geometrically, the margin is the distance between the two closest projections. As illustrated in Figure 3.3, the distance between object 1 and object 2 by using the weight vector $\overrightarrow{w_1}$ is δ_1 and the distance between object 1 and object 4 by using the weight vector $\overrightarrow{w_2}$ is δ_2. Figure 3.3 ((Joachims, 2002)) also illustrates how the weight vector affects the ordering of four objects in a two-dimension example. For a particular weight vector, the objects are ordered according to their projection into the weight vector. Therefore, the four objects are ordered (1, 2, 3, 4) and (2, 3, 1, 4) when we use $\overrightarrow{w_1}$ and $\overrightarrow{w_2}$ as the weight vector, respectively.

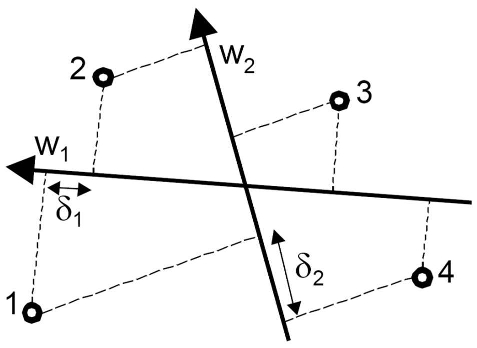


Figure 3.3: An illustrative example of the ranking of four objects by adjusting the weight vector.

Many researchers have experimented with Ranking SVM (Cao et al., 2006; Joachims, 2002; Metzler & Kanungo, n.d.; Wang et al., 2007). For instance, Joachims (2002) showed how to use Ranking SVM to exploit click-through data for optimising search engines.

3.2.3 The Listwise Approach

Though the pairwise approach has the advantage that existing classification algorithms can be directly applied, there are also some problems with this approach (Cao et al., 2007): First, the learning objective of the pairwise approach is to minimise loss during the classification of document pairs rather than the ranking of documents. Second, the pairwise approach operates under the assumption that the document pairs are generated independently and identically distributed. However, it does not always hold in many challenging tasks, such as ranking, active learning and language processing. Third, the number of generated document pairs varies largely from query to query, which can result in the loss function biased towards the queries that have a larger number of document pairs.

In order to avoid these issues, many listwise techniques for learning to rank have been proposed (Cao et al., 2007; Chakrabarti et al., 2008; Taylor et al., 2008; Xia et al., 2008; Xu & Li, 2007; Yue et al., 2007), which consider document lists instead of document pairs as the instances during the training process. In general, they can be classified into two categories (Liu, 2009): techniques in the first category perform a direct optimisation of an IR evaluation measure, with the loss function defined based on the approximation or bound of the evaluation measure, such as MAP and P@10. The second category deals with the minimisation of listwise ranking loss, in which the loss function measures the difference between the permutation given by the hypothesis and the ground truth permutation.

3.2.3.1 Direct Optimisation of IR Evaluation Measures

SoftRank Typical IR evaluation metrics are only dependent on the ranks of documents instead of document relevance scores. If we make a slight change to the parameter of a ranking function, the document relevance scores would be changed smoothly. However, the ranks of these documents may not change if no swap between documents. In this case, we cannot see any difference before and after the parameter modification according to the IR evaluation scores. In other words, the IR evaluation metrics are non-smooth with respect to the parameter of the ranking function.

However, many machine learning algorithms require the gradient of a training objective in order to optimise the parameters of a ranking function. In order to find a smooth proxy objective for the optimisation, Taylor et al. (2008) proposed a new family of training objectives that are derived from the rank distributions of documents, called SoftRank. In particular, they presented a smoothed approximation to the original IR evaluation measure: Normalised Discounted Cumulative Gain (NDCG), called SoftNDCG.

The SoftRank algorithm contains the following steps:

- **Smoothing scores**: For a given query q and a set of retrieved documents $D = \{d_j\}_{j=1}^{m}$, each document's relevance score s_j is no longer treated as a deterministic value but as a smoothed score distribution. In addition, the smoothed score distribution is modelled by a Gaussian score distribution whose variance is σ_s and mean is s_j:

$$p(s_j) = N(s_j | f(d_j), \sigma_s^2) \tag{3.12}$$

- **From score to rank distribution**: Due to the randomness of the ranking scores of the documents, every document has a probability of being ranked at a higher position than another. Given a pair of documents: d_u and d_v, the probability that document d_u beats document d_v is the difference of two Gaussian random variables, given as follows:

$$p_{u,v} = \int_0^{\infty} N(s|f(d_u) - f(d_v), 2\sigma_2^2)ds \tag{3.13}$$

This quantity represents the fractional number of times that document d_u ranks higher than document d_v on repeated pairwise instances from the two Gaussian score distributions.

Based on the above pairwise probabilities, a rank distribution can be derived in an iterative manner. Assuming a document d_u to be added into a ranked list which already contains document d_j, there are two possibilities: either d_u beats d_j or d_j ranks higher. For the first case, the probability of being in rank r at this iteration is equal to the probability of being in rank $r-1$ on the previous iteration. As to the second case, d_u leaves d_j unchanged and the probability of being in rank r is the same as it was in the last iteration. The sum of the two parts is presented as follows:

$$p_j^u(r) = p_j^{u-1}(r-1)p_{u,j} + p_j^{u-1}(r)(1 - p_{u,j}) \tag{3.14}$$

- **SoftNDCG**: By taking the NDCG evaluation measure as an example, SoftRank computes the expectation of NDCG with respect to the rank distribution as follows:

$$SoftNDCG = \frac{1}{Z_m}\sum_{j=1}^{m}(2^{y_j} - 1)\sum_{r=0}^{m-1} d(r)p_j(r) \tag{3.15}$$

where $d(r) = \frac{1}{log(2+r)}$, y_j is the relevance label for document d_j, Z_m is the maximum value of $\sum_{j=1}^{m}(2^{y_j} - 1)\sum_{r=0}^{m-1} d(r)p_j(r)$, which is obtained when the documents are optimally ordered. To learn a ranking function f by maximising SoftNDCG, a neural network and gradient descent are used as the model and optimisation algorithm, respectively.

Experimental results in (Taylor et al., 2008) showed that SoftRank is a very good way of optimising NDCG, and it is possible to achieve state of the art test set NDCG results by optimising a soft NDCG objective on the training set with a different discount function.

AdaRank Several learning to rank techniques, such as RankNet (Burges et al., 2005), RankBoost (Freund et al., 2003), and Ranking SVM (Joachims, 2002), learn a ranking function for a specific task by optimising a selected loss function. However, for these, the loss function may only be loosely related to standard IR evaluation measures. This could result in the obtained ranking function deviating from the target evaluation measure and producing poor retrieval performance.

To avoid this issue, Xu & Li (2007) proposed the AdaRank algorithm, which is a boosting-based method and employs an exponential loss function based on IR evaluation metrics. Similar to the AdaBoost algorithm (Freund & Schapire, 1995), AdaRank can focus more on the difficult queries during the construction of a ranking function.

AdaRank considers each document feature as a weak ranker and assigns an equal weight to each query at the beginning. At each round, the weak ranker with the best overall performance for a considered evaluation measure (e.g. mean average precision) over the training query set is added into the final ranking function. After a weak ranker is added, query weights are updated based on the retrieval performance of the new obtained ranking function, with the aim of making future weak learners focus more on the queries that previous weak rankers inaccurately ranked.

For a given training set that contains m queries:$\{q_i\}_{i=1}^{m}$, the evaluation of a weak ranker h_t and a ranking model f_t on a training query are denoted as $E(h_t, q_i)$ and $E(f_t, q_i)$, respectively. The learning algorithm for AdaRank is presented in Algorithm 2.

Automatic Feature Selection In order to relax the work of manual document feature selection in IR, Metzler (2007) proposed an Automatic Feature Selection (AFS) method, which also directly optimises an IR evaluation measure. It is an interactive algorithm, the general idea of which is as follows: start with an empty ranking function; then, at each round, the best performing document feature for the target evaluation measure (e.g. Mean Average Precision) in the training data set is selected; this selected feature is then removed from the candidate document feature set and added into the ranking function with an appropriate weight, which can be obtained by using learning to rank techniques (Burges et al., 2005;

```
**Algorithm**: AdaRank
**Input**: $\{q_i\}_{i=1}^{m}$
**Initialise**: $P_1(i) = \frac{1}{m}$
**For** $t = 1, ..., T$:
    • Create a weak ranker $f_t$ based on distribution $\mathbf{P}_t$.
    • Choose $\alpha_t$
      $\alpha_t = \frac{1}{2} \cdot log \frac{\sum_{i=1}^{m} P_t(i)\{1+E(h_t,q_i)\}}{\sum_{i=1}^{m} P_t(i)\{1-E(h_t,q_i)\}}$
    • Create $f_t$
      $f_t = \sum_{k=1}^{t} \alpha_k \cdot h_k$
    • Update $\mathbf{P}_{t+1}$
      $P_{t+1}(i) = \frac{exp\{-E(f_t,q_i)\}}{\sum_{j=1}^{m} exp\{-E(f_t,q_i)\}}$
**End For**
**Output**: $f = f_T$
```

Algorithm 2: The algorithm of AdaRank.

Joachims, 2005; Metzler & Croft, 2007); the algorithm terminates after a finite number of iterations, or when none of the remaining features improve retrieval effectiveness. The feature selection algorithm is presented in Algorithm 3.

Compared to the AdaRank algorithm, there are three main differences:

- For the AdaRank algorithm, the document feature (or weak ranker) that has been picked out from the candidate feature set can be re-used in the next iteration, while for the AFS method, the selected feature is removed from the candidate feature set.

- AdaRank uses boosting to focus on improving difficult queries, whereas queries are treated equally in the AFS method.

- AdaRank is more efficient than the AFS method: in AdaRank, the weight of a feature is automatically calculated according to the feature's performance on the training dataset after each round; for AFS, the weight of each candidate feature is obtained using a learning to rank technique at each iteration.

```
Algorithm: Automatic Feature Selection
Input: $F = \{f_i\}_{i=1}^{m}$
Initialise: $t \leftarrow 0$; $R_t \leftarrow \{\}$
While $E(R_t) - E(R_{t-1}) > \epsilon$ do
    • Choose $\alpha_i$
    • $f^* = arg\ max_{f_i} E(R_t + \alpha_i \cdot f_i)$
    • Update $R_{t+1} = R_t + \alpha_i \cdot f^*$
    • $F \leftarrow F - f^*$
    • $t \leftarrow t + 1$
End While
Output: $R_t$
```

Algorithm 3: The algorithm of automatic feature selection.

3.2.3.2 Minimisation of Listwise Ranking Loss

In statistics, several famous models have been previously proposed to represent a probability distribution on permutations, such as the Luce model (Luce, 1959) and the Mallows model (Mallows, 1975). By using similar probability distributions, Cao et al. (2007) proposed a probabilistic method to calculate a listwise ranking loss, called ListNet.

For a given query q and a set of retrieved documents $\{d_j\}_{j=1}^{m}$, each document's relevance score is computed by a scoring function f:$\{s_j\}_{j=1}^{m}$, where $s_j = f(d_j)$. By using the Luce model, the probability of permutation π given the list of scores s is defined as:

$$P_s(\pi) = \prod_{j=1}^{m} \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^{m} \phi(s_{\pi(k)})} \tag{3.16}$$

where $s_{\pi(j)}$ denotes the score of object at the j-th position of the permutation π. ϕ is a transformation function, which can be linear, exponential or sigmoid.

To better understand the equation, we now provide an example. For a given query q, there are three retrieved documents: A, B and C. In addition, $f(A) = 5$, $f(B) = 3$ and $f(C) = 1$. The probability of permutation $\pi = (ABC)$ is equal to the product of the following three probabilities (i.e. $P_\pi = P_1 \cdot P_2 \cdot P_3$).

- P_1: the probability of document A being ranked at the top position in π, which is computed as follows:

$$P_1 = \frac{\phi(s_A)}{\phi(s_A) + \phi(s_B) + \phi(s_C)} = \frac{5}{5+3+1} = 0.5556 \tag{3.17}$$

- P_2: the probability of document B being ranked at the second position given that document A has been ranked at the top position, given as follows:

$$P_2 = \frac{\phi(s_B)}{\phi(s_B) + \phi(s_C)} = \frac{3}{3+1} = 0.75 \tag{3.18}$$

- P_3: the probability of document C being ranked on the third position given that documents A and B have been ranked in the top two positions, computed as follows:

$$P_3 = \frac{\phi(s_C)}{\phi(s_C)} = \frac{1}{1} = 1 \tag{3.19}$$

With the Luce model, for a given query, ListNet first computes the permutation probability distribution based on the scores assigned by scoring function f. Therefore, $P_\pi(ABC) = P_1 \cdot P_2 \cdot P_3 = 0.5556 \cdot 0.75 \cdot 1 = 0.4167$. Then, it computes another permutation probability distribution based on the ground truth r. Assuming the ground truth r of the relevance degree of each document to q as: $r(A) = 3$, $r(B) = 2$ and $r(C) = 1$, then the probability of the permutation by using the ground truth is computed as:

$$P_\pi(ABC) = \frac{3}{3+2+1} \cdot \frac{2}{2+1} \cdot \frac{1}{1} = 0.3333 \tag{3.20}$$

With the obtained two permutation probability distributions, the loss function in ListNet is defined as the difference between the two distributions:

$$L(y, z) = -\sum_{\forall g \in G} P_y(g) log(P_z(g)) \tag{3.21}$$

where G is the permutation set, $P_y(g)$ and $P_z(g)$ are the permutation probability distributions obtained by using f and t, respectively.

	MAP			
	TREC2003	TREC2004	TREC2007	TREC2008
Ranking SVM	0.5366	0.4193	0.4641	0.4752
AdaRank	0.5977	0.5062	0.4537	0.4766
AFS	0.6145	0.5079	0.4596	0.4784
Upper Bound	**0.6933** $\star * \dagger$	**0.5744** $\star * \dagger$	**0.5057** $\star * \dagger$	**0.5226** $\star * \dagger$

Table 3.1: The highest score in each column is highlighted in bold and scores that are statistically better than *RankingSVM*, *AdaRank*, and *AFS* are marked with $\star$, $*$, and $\dagger$, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$).

3.2.4 Summary

In this section, several different learning to rank techniques were introduced, including the pointwise, pairwise and listwise approaches. However, the ranking functions learned from these learning to rank techniques are usually systematically applied to all queries. This ignores the fact that different ranking functions favour different queries, which was observed in (Peng et al., 2010).

Table 3.1 shows the retrieval performance of three LTR techniques, namely Ranking SVM, AdaRank and the AFS method, on four different datasets[1]. Moreover, Upper Bound is achieved by manually selecting the most effective ranking function on a per-query basis. From this table, it is clear that the retrieval performance can be significantly enhanced if the most appropriate ranking function is applied for each query. This observation suggests that different ranking functions do favour different queries and that the appropriate selective application of a ranking function could enhance the retrieval performance. To solve this issue, several selective retrieval approaches are discussed in the remaining of this chapter.

3.3 Query Dependent Ranking

Geng et al. (2008) proposed a query-dependent ranking approach. For each given query, they employ a specific ranking function, which is obtained by applying a LTR technique (e.g. Ranking SVM) on a training query set. This training

[1]The detailed settings can be found in (Peng et al., 2010).

query set is dependent on the given query, which can be identified by using a classification technique (i.e. K-Nearest Neighbour) based on a query feature. The query feature used in their work is the mean of the document feature scores of the top retrieved documents, which can be obtained by a reference model (e.g. BM25), given as follows:

$$score(q, r_i) = \frac{\sum_{\phi=1}^{n} rel(d_\phi)}{n} \tag{3.22}$$

where n is the number of the top retrieved documents returned by the ranking function r_i for a given query q, and $rel(d_\phi)$ is the document relevance score of a document d at position ϕ of the document ranking list.

In particular, they proposed three different versions of the query dependent ranking technique with the aim of reducing time complexity: KNN Online, KNN Offline-1 and KNN Offline-2, which are explained in detail in the following.

3.3.1 The KNN Online Algorithm

The KNN Online algorithm contains two processing stages:

- Offline pre-processing: Compute the query feature score for each training query q_i by using Equation (3.22).
- Online training and testing: 1) For a given test query q, find its k nearest neighbouring queries from the training query set, according to its query feature score. 2) Obtain a ranking function by applying a LTR technique (e.g. Ranking SVM) on the identified neighbour query set. 3) Apply this obtained ranking function to the given test query q and obtain the document ranking list.

An illustration of the KNN Online algorithm is presented in Figure 3.4. In this figure, the square symbol stands for the test query q, each triangle denotes a training query, and the triangles that are located in the circle stand for the neighbouring queries of q.

The KNN Online algorithm is easy to follow. However, the main drawback of the algorithm is efficiency, as it needs to conduct the training of the ranking function for each test query.

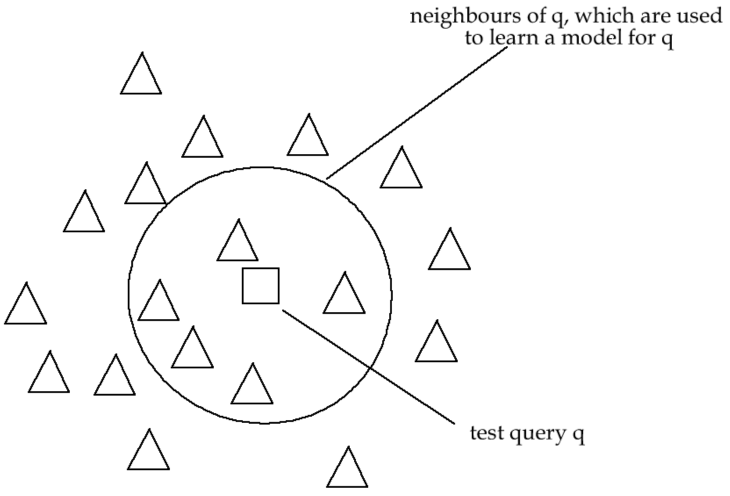


Figure 3.4: An illustration of the KNN Online algorithm for building a document ranking function.

3.3.2 The KNN Offline-1 Algorithm

To reduce the time complexity, Geng et al. (2008) proposed another version of the query dependent ranking technique, called KNN Offline-1, which also contains two processing stages:

- Offline training: 1) Compute the query feature score for each training query q_i by using Equation (3.22). 2) For each training query q_i, find the k nearest neighbouring queries from the training query set, denoted as $N_k(q_i)$. 3) Learn a ranking function $r(q_i)$ for each q_i by applying a LTR technique on its corresponding neighbour query set.
- Online testing: 1) Compute the query feature score for the test query q. 2) Find k nearest neighbour queries from the training query set, denoted as $N_k(q)$, according to the query feature score. 3) Find the most similar training set $N_k(q*)$ by using Equation (3.23). 4) Apply the ranking function $r(q*)$ to the test query q and obtain the document ranking list.

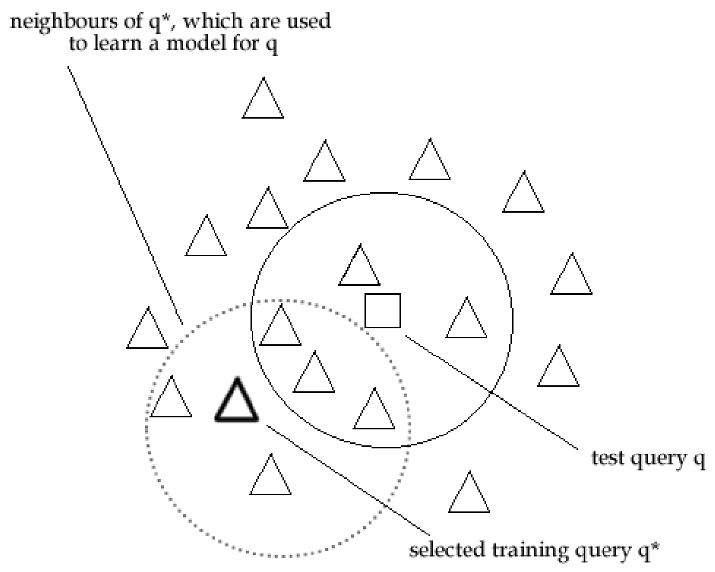

neighbours of q*, which are used to learn a model for q
test query q
selected training query q*

Figure 3.5: An illustration of the KNN Offline-1 algorithm for building a document ranking function.

$$N_k(q*) = \arg\max_{N_k(q_i)} |N_k(q_i) \cap N_k(q)| \tag{3.23}$$

where $|N_k(q_i) \cap N_k(q)|$ denotes the number of common queries shared by $N_k(q_i)$ and $N_k(q)$.

An illustration of the KNN Offline-1 algorithm is presented in Figure 3.5. In this figure, the square symbol stands for the test query q, each triangle denotes a training query, and the bold triangle stands for the selected training query based on Equation 3.23. The triangles that are located in the solid-line circle and dotted-line circle stand for the neighbouring queries of q and $q*$, respectively.

Compared to KNN Online, the KNN Offline-1 algorithm saves the online training time. However, it introduces additional computation for finding the most similar training query set, namely the third step of the *Online testing* stage.

3.3.3 The KNN Offline-2 Algorithm

In order to avoid the additional computation time, Geng et al. (2008) proposed the KNN Offline-2 algorithm, given as follows:

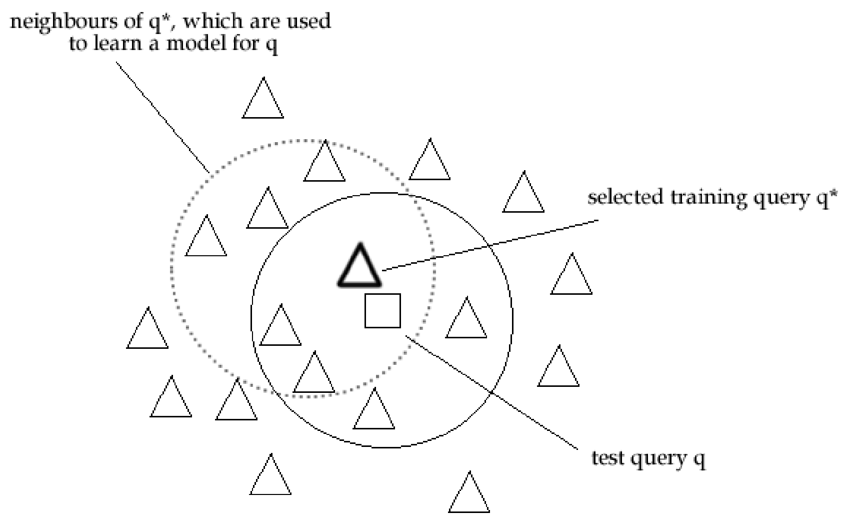

neighbours of q*, which are used to learn a model for q
selected training query q*
test query q

Figure 3.6: An illustration of the KNN Offline-2 algorithm for building a document ranking function.

- Offline training: 1) Compute the query feature score for each training query q_i by using Equation (3.22). 2) For each training query q_i, find the k nearest neighbouring queries from the training query set, denoted as $N_k(q_i)$. 3) Learn a ranking function $r(q_i)$ for each q_i by applying a LTR technique on its corresponding neighbour query set.
- Online testing: 1) Compute the query feature score for the test query q. 2) Find the *single* nearest neighbouring query $q*$ from the training query set, according to the query feature score. 3) Apply ranking function $r(q*)$ to the test query q and obtain the document ranking list.

An illustration of the KNN Offline-2 algorithm is presented in Figure 3.6. In this figure, the square symbol stands for the test query q, each triangle denotes a training query, and the bold triangle stands for the selected training query $q*$ according to the query feature score. The triangles located in the solid-line circle and dotted-line circle stand for the neighbouring queries of q and $q*$, respectively.

In comparison with the KNN Offline-1 algorithm, we note that the KNN Offline-2 algorithm actually finds the nearest neighbour query $q*$ of q, directly according to the query feature score.

Extensive experiments on a large dataset, sampled from a commercial search engine, have shown the effectiveness of this approach (Geng et al., 2008). However, this work only investigated the selective application of a ranking function obtained from a single learning to rank technique and a fixed set of document features. Hence, the effectiveness of the query-dependent ranking approach is not clear when there is more than one LTR technique and the number of features is varied. Moreover, the use of the mean of the feature scores from the top retrieved documents simply ignores the importance of the distribution of the feature scores, which has been effectively used in many applications (Manmatha et al., 2001; Peng & Ounis, 2009). For example, Manmatha et al. (Manmatha et al., 2001) use the relevance score distribution to estimate the effectiveness of a search engine.

3.4 Selective Collection Enrichment

A query performance predictor aims to predict whether a given query is an “easy” or a “difficult” query for a given document collection (Hauff et al., 2008). This kind of prediction mainly relies on the statistics of the collection for this given query, such as query term frequency in the collection and the number of documents containing the query term. Several query performance predictors, including both pre-retrieval and post-retrieval predictors, were introduced in Section 2.4.3.

Using query performance predictors, Peng et al. (2009) proposed a decision mechanism to decide whether or not to apply collection enrichment on a per-query basis. The approach is based on the predicted performance score of a given query on the local and external resources. In particular, the decision mechanism applies collection enrichment if and only if the predicted query performance score obtained on the external resource is higher than a threshold, as well as the predicted query performance score obtained using the local resource. Table 3.2 summarises the proposed decision mechanism for the selective application of collection enrichment.

It is of note, however, that for some query performance predictors, the lower the predictor score for the external resource, the higher the query performance on that resource is predicted to be, and hence the more beneficial collection

$score_L > T$	$score_E > T$	$score_L > score_E$	Decision
True	True or False	True	local
True or False	True	False	external
False	False	True or False	disabled

Table 3.2: The decision mechanism for the selective application of CE. $score_L$ and $score_E$ denote the predicted performance of a given query on the local and external resources, respectively. *local*, *external* and *disabled* in the column *Decision* indicate expanding the initial query on the *local* resource, *external* resource and disabling the expansion, respectively.

enrichment using that resource should be. For example, for the clarity score predictor (Cronen-Townsend et al., 2002), a lower query performance score on the external resource means a higher similarity between the query language model and the external collection's language model, suggesting that applying CE could bring more useful expansion terms. Hence, by using the query performance predictor to selectively apply collection enrichment, the nature of the used predictor is taken into account when comparing the query performance scores.

The selective collection enrichment approach mainly relies on the statistics of a given query in a corresponding collection rather than a given ranking function. Therefore, the selective collection enrichment approach may not be applicable to the selective application of a ranking function, as these statistics are invariant with changes in the ranking function.

3.5 Selective Web Information Retrieval

Plachouras (2006) proposed a novel framework for selective Web information retrieval. They employ a Bayesian decision mechanism to selectively apply a retrieval approach under the assistance of an experiment ε, which extracts a feature from a sample of document set.

Among k candidate retrieval approaches, namely $r_1, ..., r_k$, the probability of retrieval approach r_i being the most appropriate for a given experiment output o, according to the Bayes decision rule, is given as follows:

$$P(r_i|o) = \frac{P(r_i) \cdot P(o|r_i)}{P(o)} \tag{3.24}$$

where $P(r_i)$ is the prior probability of r_i being the most appropriate retrieval approach and is set equal to the proportion of the number of training queries, for which the retrieval approach r_i is the most effective. $P(o|r_i)$ is the probability of the outcome of experiment ε being o when the most effective retrieval approach is r_i. This probability is computed by estimating the density of the outcome values of the experiment ε on the training dataset, for which the retrieval approach r_i is the most effective. In addition, $P(o)$ is defined as follows:

$$P(o) = \sum_{i=1}^{k} P(r_i) \cdot P(o|r_i) \tag{3.25}$$

Furthermore, they estimate the expected loss of applying the retrieval approach r_i as follows:

$$E[l(r_i)] = \sum_{j=1}^{k} l(r_i, r_j) \cdot P(r_j|o) \tag{3.26}$$

where $l(r_i, r_j)$ is the loss of applying retrieval approach r_i while the most effective retrieval approach for the given query is r_j, which is defined as follows:

$$l(r_i, r_j) = \frac{rank(r_i, r_j)}{n-1} \tag{3.27}$$

where $rank(r_i, r_j)$ is the rank of the retrieval approach r_i among the n candidate retrieval approaches.

Finally, the retrieval approach with the minimum expected loss $E[l(r_i)]$ is selected for a given query.

Several different experiments ε were defined in their work (Plachouras, 2006) and are classified into two categories, namely the score-independent and score-dependent experiments. The score-independent experiments do not take into account of the relevance scores that are assigned to documents. Instead, they count the number of documents with at least one, or all query terms, the occurrences of query terms in a document and the number of documents that belong to the same URL domain. The score-dependent experiments are based on estimating the usefulness of the hyperlink structure in a sample of the retrieved document set.

Extensive experiments showed that the selective Web IR approach is effective when there are only two candidate retrieval approaches (Plachouras, 2006).

	Description
1	information in the query itself: the length of the query, the number of acronyms in the query and the number of stop words.
2	the inverse document frequency of the query terms.
3	the BM25 scores in the top 2000 returned documents for each query.
4	the number of the matched titles in the top 20 documents returned by the BM25 weighting model.
5	the number of the matched URLs in the top 20 documents returned by the BM25 weighting model.
6	the number of the matched anchors in the top 20 documents returned by the BM25 weighting model.
7	the proximity score in the top 20 documents returned by the BM25 weighting model.
8	the URL depth of the top 2 documents by the BM25 weighting model.

Table 3.3: The description of the query features used in the classification process for query type detection.

However, the retrieval performance obtained using this approach only improved slightly and actually decreased when more than two candidate retrieval approaches are used. This is because, in the selective Web IR framework, the higher number of retrieval approaches require more queries for the training of the Bayesian decision mechanism.

3.6 Query Type Prediction

A query classification task was introduced in the TREC 2004 Web track (Craswell & Hawking, 2004), which aims to classify a mixed set of queries into three different types, namely homepage finding, named page finding and topic distillation. In addition, Plachouras et al. (2004) showed that by applying different retrieval approaches for different query types, the obtained retrieval performance can be enhanced compared to the uniform application of one retrieval approach to all query types.

Several different approaches were proposed for predicting the query type for a given query (Song et al., 2004; Yang et al., 2004). Yang et al. (2004) proposed the use of a linguistic classifier to predict query type. The proposed linguistic classifier uses a set of heuristic linguistic features, which are identified from the analysis of the training dataset. For example, Yang et al. (2004) noted that queries that ended with uppercase letters tend to be the homepage finding query, queries that contain 4-digit year are more likely to be the named page finding query, and the topic distillation queries are shorter in general compared to the homepage finding and named page finding queries. In addition, they also identify some word cues for the named page finding queries (e.g. about, annual and report) and the homepage finding queries (e.g. home, welcome, office and bureau).

Song et al. (2004) proposed to use a classification technique to predict query type. In addition, several different query features were extracted, shown in Table 3.3, for the classification process. Their query classification technique is a two-stage process: 1) classify all queries into two categories: the topic distillation query and the homepage finding and named page finding query. 2) classify the latter into two categories: the homepage finding query and the named page finding query.

For these discussed query type prediction approaches, there are two main issues: First, the accuracy of the state-of-the-art query type prediction approaches is not high. For example, the highest accuracy of query type prediction between homepage finding and named page finding is around 68% (Craswell & Hawking, 2004). Second, though the query type-based retrieval strategy can enhance the retrieval performance, when comparing it with the systematic application of a retrieval strategy to all queries. Some queries of the same type benefit from having different retrieval strategies applied (Peng & Ounis, 2009).

3.7 Summary

This chapter described three different groups of the learning to rank (LTR) techniques, which are used to build a ranking function and have been widely used in IR systems. As the learned ranking functions are usually systematically applied to all queries, which ignores the fact that different ranking functions favour different queries, several selective retrieval approaches were discussed to solve this issue. However, the approaches described have various drawbacks, which prevent the wide deployment of the selective application of a specific ranking function for a given query.

The query dependent ranking approach is only investigated in a single learning to rank technique and a fixed set of document features, its effectiveness is not clear when there is more than one learning to rank technique and the number of document features is varied. Moreover, the use of the mean of the document feature scores from the top retrieved documents ignores the importance of the distribution of the document relevance scores, which has been effectively used in many applications (He et al., 2009; Manmatha et al., 2001).

The retrieval performance obtained using the selective Web IR approach only improved slightly and actually decreased when more than two candidate retrieval approaches are used. This is because, in the selective Web IR framework, the higher number of candidate retrieval approaches require more queries for training the Bayesian decision mechanism.

For the query performance predictor-based approach, it may not be applicable to the selective application of a ranking function. This is due to the fact that the predictors mainly rely on the statistics of the collection and these statistics are invariant to changes in the ranking function. As for the query type prediction approach, the accuracy of query type prediction is not high (Craswell & Hawking, 2004) and queries of the same type may benefit from having different retrieval approaches applied (Peng & Ounis, 2009). In the next chapter, we present our proposed learning to select framework, which is agnostic to the problems that inherent with these introduced selective retrieval approaches.

Chapter 4

Learning to Select

4.1 Introduction

The previous chapter presented four different selective retrieval approaches. In Section 3.3, we discussed the query dependent ranking approach. However, this approach's effectiveness is not clear when there is more than one Learning to Rank (LTR) technique and the number of document features is varied. Moreover, the use of the mean of the document feature scores from the top retrieved documents ignores the importance of the distribution of the document relevance scores, which has been effectively used in many applications (He et al., 2009; Manmatha et al., 2001). For example, Manmatha et al. (2001) used the relevance score distribution to estimate the effectiveness of a search engine.

In Section 3.4, the query performance predictor-based approach was discussed. However, this approach may not be applicable to the selective application of a ranking function. This is due to the fact that the predictors mainly rely on the statistics of the collection, such as query term frequency in the collection and the number of documents containing the query terms. Hence, the query performance predictor-based approach may not be applicable to the selective application of ranking functions, as these collection statistics are invariant to changes in the ranking function.

The selective Web IR approach (Section 3.5) has shown its effectiveness when there are only two candidate retrieval approaches. However, the retrieval performance obtained using this approach only improved slightly and actually decreased when more than two candidate retrieval approaches are used. As for the query type prediction approach (Section 3.6), there are two main issues: first, the accuracy of the existing query type prediction approaches is not high (Craswell & Hawking, 2004); second, queries of the same type may benefit from having different retrieval approaches applied (Peng & Ounis, 2009).

In order to selectively apply an appropriate ranking function from a large set of candidate ranking functions, we propose a novel Learning to Select (LTS) framework, which is agnostic to the number of ranking functions, as well as to the type of the queries. A central concept of this framework is that the effectiveness of a ranking function for a given unseen query can be estimated based on its performance on similar queries which have already been seen. To identify similar queries, we propose a neighbouring query search approach. This approach employs a classification algorithm (e.g. k-Nearest Neighbour) to find similar already seen queries for a given unseen query, by using a query feature. In particular, we propose a novel query feature, which takes into account the distribution of the document relevance scores. This new feature is based on a divergence measure, which is used to determine the extent to which a document ranking function alters the scores of an initial ranking of documents.

The remainder of this chapter is organised as follows. Section 4.2 introduces the proposed learning to select framework for selectively applying an appropriate ranking function on a per-query basis. Section 4.3 shows how to use the learning to select framework to select multiple appropriate document features for building a ranking function, on a per-query basis. The description of each component of the learning to select framework is presented in Section 4.4. An illustrative example of the learning to select framework is presented in Section 4.5. Section 4.6 compares the learning to select framework with the existing selective retrieval approaches. Finally, a summary of this chapter is presented in Section 4.7.

4.2 The Learning to Select Framework

A document ranking function created by a learning to rank technique is based on the assumption that the training dataset is representative of unseen queries. However, some queries may benefit from applying different ranking functions. We believe that the effectiveness of a ranking function for an unseen query can be estimated based on similar training queries. A divergence measure can be used to determine the extent to which a document ranking function alters the scores of an initial ranking of documents. We propose that this divergence can be used to identify similar training queries. In this case, a ranking function, which performs well for training queries that have a similar divergence to the unseen query, will also perform well on the unseen query.

```
LTS[$q'$, $R$, $Q$, $\mathbf{f}(r_i, Q)$]
1: for $i = 1, ..., n$ do
2:     With the ranking function $r_i$, compute its corresponding query feature score
       $f(r_i, q')$
3:     Based on the computed $f(r_i, q')$, find neighbouring query set $Q'_i$ from
       $\mathbf{f}(r_i, Q)$
4:     Evaluate the performance of the ranking function $r_i$ on the obtained neigh-
       bouring query set:
       $P(Q'_i, r_i) = \frac{\sum_{q_\phi \in Q'_i} P(q_\phi, r_i)}{|Q'_i|}$
5: end for
Return $r_i^*(q') = \arg\max_{r_i} P(Q'_i, r_i)$
```

Algorithm 4: The learning to select framework.

Based on the above general idea, we present the detailed learning to select algorithm as follows:

- Initially, on a training dataset, we have a set of queries $Q = \{q_1, q_2, ..., q_m\}$ and a set of candidate ranking functions $R = \{r_1, r_2, ..., r_n\}$. For each query q_j, we estimate a query feature score for each ranking function r_i, denoted as $f(r_i, q_j)$. For all training queries Q, each ranking function r_i's query feature scores set is denoted $\mathbf{f}(r_i, Q) = \{f(r_i, q_1), ..., f(r_i, q_m)\}$.

- Next, in response to an unseen query q', for each ranking function r_i, we first estimate a query feature score $f(r_i, q')$, then employ a neighbouring query search technique to identify a set of the most similar queries (Q'_i) from $\mathbf{f}(r_i, Q)$ according to the distance between queries using the query feature. Each identified similar query corresponds to a query q_ϕ.

- Let $0 \leq E(q_\phi, r_i) \leq 1$ be the outcome of an evaluation measure (see Section 2.7) calculated on the ranking function r_i for the similar training query q_ϕ. The performance of ranking function r_i on this test query q' is predicted based on the performance of r_i on the identified similar query set Q'_i of q':

$$P(r_i, q') \simeq P(r_i, Q'_i) = \frac{\sum_{q_\phi \in Q'_i} E(q_\phi, r_i)}{|Q'_i|} \tag{4.1}$$

- Finally, we apply the ranking function r_i for the query q' that has the highest retrieval performance on the neighbouring query set:

$$r_i^*(q') = \arg\max_{r_i} \frac{\sum_{q_\phi \in Q'_i} E(q_\phi, r_i)}{|Q'_i|} \tag{4.2}$$

The LTS algorithm, written in pseudo-code, can also be found in Algorithm 4.

4.3 Selecting Multiple Document Features

Algorithm 4 shows how to select an appropriate ranking function from a number of candidate ranking functions for a given query. The candidate ranking functions are usually built on the same set of document features. However, in some cases, we need to select multiple appropriate document features for building a ranking function. For example, with two document features (e.g., *PageRank* and *URL type*) and two different queries (e.g., q'_A and q'_B), assume that the best retrieval performance for q'_A is achieved by a ranking function that is built on both document features while the best retrieval performance for q'_B is achieved by a ranking function that is built on *PageRank* only. In this case, to achieve the overall best retrieval performance, a system should selectively choose multiple appropriate document features for building a ranking function.

To adapt our proposed learning to select framework to take into account this case, only one extra step is required: for a given document feature set, we regenerate this set by using possible combinations among the included document features. In other words, for a document feature set that contains n document features, there are $2^n - 1$ possible combinations (excluding the empty combination).

In the example above, there are $2^2 - 1 = 3$ possible combinations: *PageRank*, *URL type* and *PageRank+URL type*. A ranking function can be learned by applying a learning to rank technique to each combination. In this case, selecting multiple appropriate document features from a document feature set that contains n document features is equivalent to selecting an appropriate ranking function from $2^n - 1$ candidate ranking functions.

4.4 The Components of Learning to Select

The proposed learning to select framework contains two main components:

- **The query feature component**, which is used to represent the characteristics of a given query. For example, the mean of the document relevance scores, as used by Geng et al. (2008).
- **The identifying neighbouring queries component**, which is a technique that is used to identify the most similar queries from a training dataset, for a given unseen query. An example of an applicable algorithm is the K-Nearest Neighbour (KNN) search (Cover & Hart, 1967).

The details of each component are presented in the remainder of this section.

4.4.1 Query Features

As presented in Section 3.3, Geng et al. (2008) used the mean of document relevance scores as a query feature to identify neighbouring queries. However, this query feature ignores the distribution of the document relevance scores, which was shown to be important in many applications (He et al., 2009; Manmatha et al., 2001). For example, (He et al., 2009) used the distribution of the document relevance scores to identify a suitable document ranking for blog opinion retrieval.

By plotting the top 1000 retrieved documents' relevance scores distribution in Figure 4.1, we observe two different distributions that are produced by two different ranking functions on a same set of documents. In Figure 4.1, the ranking function that is used to obtain the initial ranking of documents for a given query is called the *base ranking function*. Any other ranking functions that may be

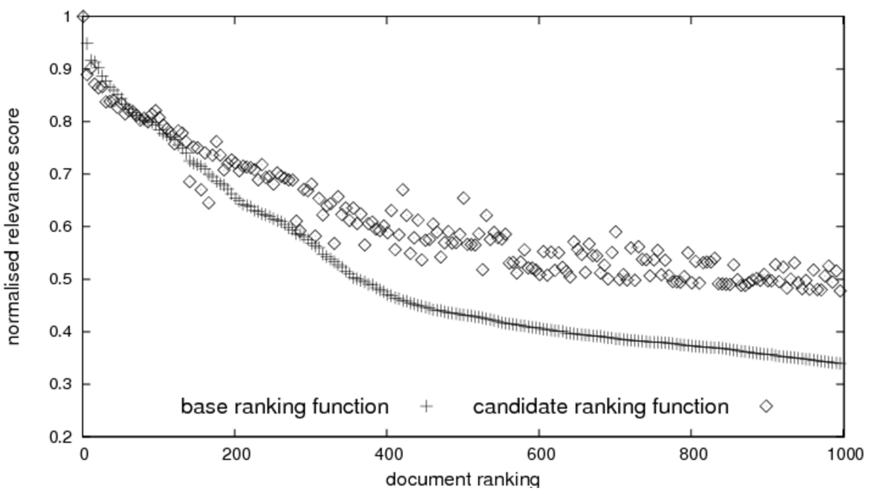


Figure 4.1: Score distributions of the top 1000 retrieved documents, which are ranked according to their relevance scores in the base ranking function.

applied are called *candidate ranking functions*. They assign different document relevance scores to the same documents as retrieved by the base ranking function. A divergence measure can be used to determine the extent to which a document ranking function alters the scores of an initial ranking of documents.

To find the relation between the estimated divergence score and the effectiveness of a ranking function, we plot the distribution of the divergence scores (estimated between a base ranking function and a candidate ranking function) versus the relative retrieval effectiveness (obtained by conducting a subtraction between the retrieval performances, e.g. MAP, of a base ranking function and a candidate ranking function) in Figure 4.2. In particular, we use the TREC 2004 Web track data as an example, which contains 225 queries. We divide the estimated divergence scores into 8 equal size bins. The X axis corresponds to the divergence score of each bin, while the Y axis corresponds to the mean of the relative retrieval effectiveness that is obtained for the queries which belong to the same bin. In addition, a standard error bar is also provided to show the uncertainty in the measure of relative retrieval performance.

From Figure 4.2, we note that the mean of the relative retrieval effectiveness decreases as the estimated divergence score increases. However, after reaching its lowest value, the mean of the relative retrieval effectiveness starts increasing as

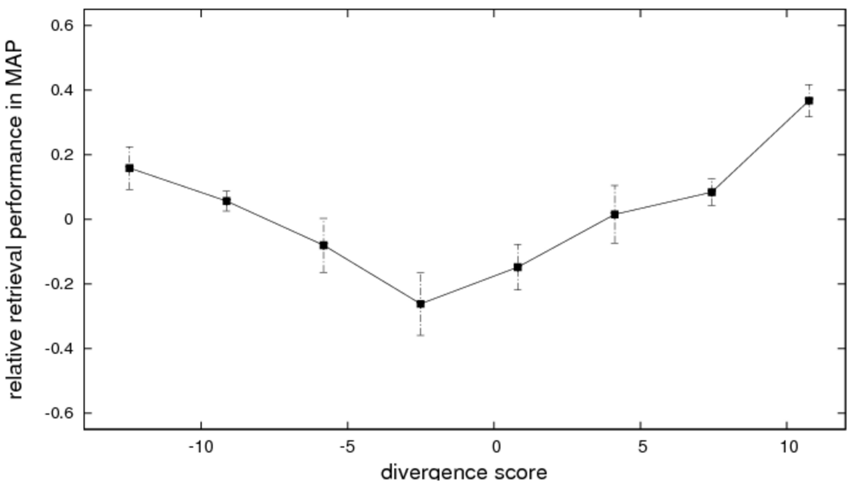


Figure 4.2: The distribution of the relative retrieval performance in MAP, with standard error bars, versus the estimated divergence score.

the divergence score increases. In particular, from the standard error bars, we observe that queries that have similar divergence scores receive a similar relative retrieval effectiveness. This observation suggests that the divergence measure can be used as a query feature to identify similar queries.

Based on this observation, in this section, we propose to use the divergence measure as a query feature. There are several different ways to estimate the divergence between two document score distributions that are obtained by using a base ranking function r_b and a candidate ranking function r_i. A commonly used divergence measure is the Kullback-Leibler (KL) (Kullback, 1997) divergence, given as follows:

$$KL(r_b||r_i, q) = \sum_{d=1}^{T} r_b(d) \cdot \log_2 \frac{r_b(d)}{r_i(d)} \tag{4.3}$$

where, for the top T retrieved documents of a given query q, $r_b(d)$ and $r_i(d)$ are the relevance scores of document d in the base ranking r_b and the candidate ranking r_i, respectively.

Another commonly used divergence measure is the Jensen-Shannon (JS) (Lin, 1991) measure, which is a symmetric version of the Kullback-Leibler divergence, calculated as the average of the KL divergence from probability distribution r_b

to r_i, and KL divergence from probability distribution r_i to r_b, given as follows:

$$\begin{aligned} JS(r_b||r_i, q) &= KL(r_b||(\frac{1}{2} \cdot r_b + \frac{1}{2} \cdot r_i), q) \\ &= \sum_{d=1}^{T} r_b(d) \cdot \log_2 \frac{r_b(d)}{\frac{1}{2} \cdot r_b(d) + \frac{1}{2} \cdot r_i(d)} \end{aligned} \tag{4.4}$$

It is easy to verify that adding a constant to the relevance scores of the documents in r_i does not change the ranking position of each document in r_i, however, this affects the divergence between r_b and r_i. For example, assuming four retrieved documents for a given query q: d_1, d_2, d_3, and d_4. The relevance scores of each document in the base ranking function r_b and a candidate ranking function r_i are given as follows:

	r_b	r_i
d_1	0.4	0.3
d_2	0.3	0.4
d_3	0.2	0.1
d_4	0.1	0.2

Then the estimated KL divergence score $KL(r_b||r_i, q)$ is equal to:

$$\begin{aligned} KL(r_b||r_i, q) &= 0.4 \times \log_2 \frac{0.4}{0.3} + 0.3 \times \log_2 \frac{0.3}{0.4} + 0.2 \times \log_2 \frac{0.2}{0.1} \\ &\quad +0.1 \times \log_2 \frac{0.1}{0.2} \\ &\approx 0.1415 \end{aligned} \tag{4.5}$$

According to the relevance scores produced by the ranking function r_i, the ranking position of the four documents ($Rank(r_i)$) is given as follows:

	r_b	r_i	$Rank(r_i)$
d_1	0.4	0.3	2
d_2	0.3	0.4	1
d_3	0.2	0.1	4
d_4	0.1	0.2	3

By deducting a constant (e.g. 0.05) to the relevance scores of the documents in r_i, we obtain another relevance score list r'_i:

	r_b	r_i	r'_i
d_1	0.4	0.3	0.25
d_2	0.3	0.4	0.35
d_3	0.2	0.1	0.05
d_4	0.1	0.2	0.15

The ranking position of the four documents according to r'_i is given below:

	r_b	r_i	r'_i	$Rank(r_i)$	$Rank(r'_i)$
d_1	0.4	0.3	0.25	2	2
d_2	0.3	0.4	0.35	1	1
d_3	0.2	0.1	0.05	4	4
d_4	0.1	0.2	0.15	3	3

which is exactly the same as the ranking position obtained by using r_i. However, the KL divergence score $KL(r_b||r'_i, q)$ changes to:

$$\begin{aligned} KL(r_b||r'_i, q) &= 0.4 \times \log_2 \frac{0.4}{0.25} + 0.3 \times \log_2 \frac{0.3}{0.35} + 0.2 \times \log_2 \frac{0.2}{0.05} \\ &\quad + 0.1 \times \log_2 \frac{0.1}{0.15} \\ &\approx 0.5460 \end{aligned} \tag{4.6}$$

In order to avoid the issue of translation invariance, we apply a score normalisation, which was proposed by Lee (1997), on each document of the rankings r_b and r_i:

$$r_N(d) = \frac{r(d) - r(min)}{r(max) - r(min)} \tag{4.7}$$

where $r_N(d)$ is the document relevance score after normalisation, $r(max)$ and $r(min)$ are the maximum and minimum document relevance scores that have been observed in the top retrieved documents from the input ranking r, and $r(d)$ is the relevance score of document d in the input ranking.

By applying Lee's score normalisation algorithm on the provided example, we obtain the following scores:

	r_{b_N}	r_{i_N}	r'_{i_N}
d_1	$\frac{0.4-0.1}{0.4-0.1}=1$	$\frac{0.3-0.1}{0.4-0.1}=0.66$	$\frac{0.25-0.05}{0.35-0.05}=0.66$
d_2	$\frac{0.3-0.1}{0.4-0.1}=0.66$	$\frac{0.4-0.1}{0.4-0.1}=1$	$\frac{0.35-0.05}{0.35-0.05}=1$
d_3	$\frac{0.2-0.1}{0.4-0.1}=0.33$	$\frac{0.1-0.1}{0.4-0.1}=0$	$\frac{0.05-0.05}{0.35-0.05}=0$
d_4	$\frac{0.1-0.1}{0.4-0.1}=0$	$\frac{0.2-0.1}{0.4-0.1}=0.33$	$\frac{0.15-0.05}{0.35-0.05}=0.33$

Based on the above normalised document relevance scores, we obtain the ranking positions of each document in both r_{i_N} and r'_{i_N}:

	r_{b_N}	r_{i_N}	r'_{i_N}	$Rank(r_{i\_N})$	$Rank(r'_{i\_N})$
d_1	1	0.66	0.66	2	2
d_2	0.66	1	1	1	1
d_3	0.33	0	0	4	4
d_4	0	0.33	0.33	3	3

The above results show that both ranking functions produce the same ordering of the documents after the score normalisation. In addition, the KL divergence score $KL(r_{b\_N}||r_{i\_N},q)$ is equal to $KL(r_{b\_N}||r'_{i\_N},q)$:

$$\begin{aligned} KL(r_{b\_N}||r_{i\_N},q) &= KL(r_{b\_N}||r'_{i\_N},q) \quad (4.8)\\ &= 1\times\log_2\frac{1}{0.66}+0.66\times\log_2\frac{0.66}{1}+0.33\times\log_2\frac{0.33}{0}\\ &\quad +0\times\log_2\frac{0}{0.33}\end{aligned}$$

In mathematics, the denominator is required to be non-zero in a division operation in order to obtain a real value. Moreover, only positive real numbers have real-valued logarithms. However, from the above equation, we observe $\log_2\frac{0.33}{0}$ and $\log_2\frac{0}{0.33}$. In order to obtain real-value scores in the divergence estimation, we modify Lee's score normalisation algorithm as follows:

$$r_N(d)=\frac{r(d)-r(min)}{r(max)-r(min)}+c \quad (4.9)$$

where c is a constant ($c>0$).

4.4.2 Identifying Neighbouring Queries

Finding neighbouring queries for an unseen query can be treated as a classification problem, as neighbouring queries are assumed to belong to the same group. In pattern recognition, several different classification algorithms have been previously proposed, which are based on either prior knowledge or the statistical information that is extracted from the patterns (Duda et al., 2000). In Figure 4.2, we observe that queries that have similar divergence scores receive a similar relative retrieval effectiveness. However, from the global distribution of the divergence scores, we note that some queries receive similar relative retrieval performances but have far different divergence scores. For example, queries with divergence scores around -10 exhibit similar relative retrieval performance with queries, whose divergence scores around 7. This observation emphasises the locality property of the queries. Hence, in the learning to select framework, we propose to use the k-nearest neighbour and k-means approaches for finding the neighbouring queries. In addition, the bin approach that was proposed in (Peng & Ounis, 2009) for classifying queries is also presented (Section 4.4.2.3).

4.4.2.1 k-Nearest Neighbour

The nearest neighbour search (NNS) approach, also known as proximity search, similarity search or closest point search, is used to find the closest points in a metric space, for a given point (Feustela & Shapiro, 1982). There are a number of variants of NNS, among which the most well-known is the k-Nearest Neighbour (KNN) classification technique. KNN is a type of instance-based learning, in which the function is only approximated locally (Cover & Hart, 1967). The KNN classification technique is one of the most fundamental classification techniques and is widely used for classifying objects when there is little or no prior knowledge about the distribution of the objects.

In the KNN classification algorithm, an object is classified by majority vote of its neighbours. The steps of the KNN algorithm are as follows (Mitchell, 1997):

1. Set the parameter k, which is the number of the nearest neighbours that need to be identified.

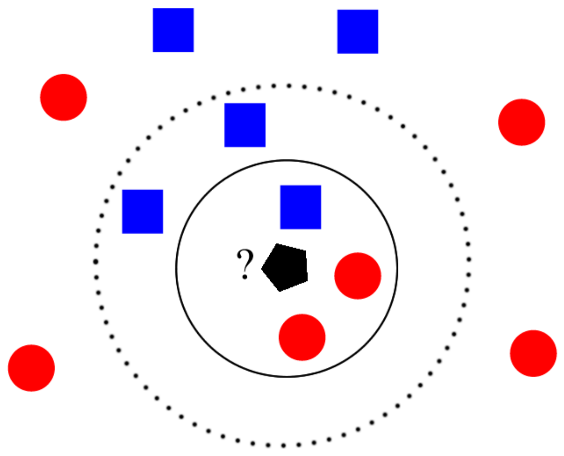


Figure 4.3: An illustration of the KNN algorithm for classifying objects.

2. Compute the distance between a target object and those that were sampled.

3. Sort the samples according to the calculated distances.

4. Collect the k nearest samples.

5. Use the simple majority of the category of the k nearest samples as the prediction value of the target object.

As illustrated in Figure 4.3, the target object (black pentagon) should be classified either as a blue square or as a red circle. It is classified as a red circle if we set $k = 3$, as the red circle is the majority among the three nearest neighbours (inside the solid-line circle). Also, it can be classified as a blue square if we set $k = 5$, as the blue square is the majority among the five nearest neighbours (inside the dotted-line circle).

To adapt the KNN algorithm for identifying similar queries in the LTS framework (the 3rd step of Algorithm 4), we modify this algorithm as follows:

1. Set the parameter k, which is the number of the similar queries that need to be identified.

2. Compute the distance between a test query q' and each training query, as the absolute difference between the feature scores of these two queries.
3. Sort the training queries according to the calculated distances.
4. Identify the k most similar (nearest) queries.
5. The identified k most similar queries are used in Equation (4.2) to decide which ranking function is the most appropriate for the test query q'.

4.4.2.2 *k*-means

While the KNN algorithm, which classifies an object based on the vote of its neighbours, the k-means classification algorithm aims to partition n objects into k groups, in which each object belongs to the group with the nearest mean (MacQueen, 1967). The steps of the algorithm are as follows:

1. Set the parameter k, which is the number of groups.
2. Generate k centroids, randomly or evenly separated within the range of the query feature scores.
3. Assign each object to its nearest centroid, the objects that are assigned to the same centroid form a group.
4. Update each centroid with the mean of the scores of the objects that are assigned to this centroid.
5. Repeat the previous two steps until some convergence criterion is met (usually when no object moves from one group to another).

Figure 4.4 illustrates the steps of the k-means algorithm. There are twelve objects (black squares) that need to be classified into three groups. In the first step, three centroids are randomly generated (namely the red circle, the green circle, and the blue circle). In the second step, each object is assigned to its nearest centroid. In this case, there is only one object that belongs to the red centroid group, six objects belong to the green centroid group and five objects belong to the blue centroid. In the third step, each centroid is updated with the

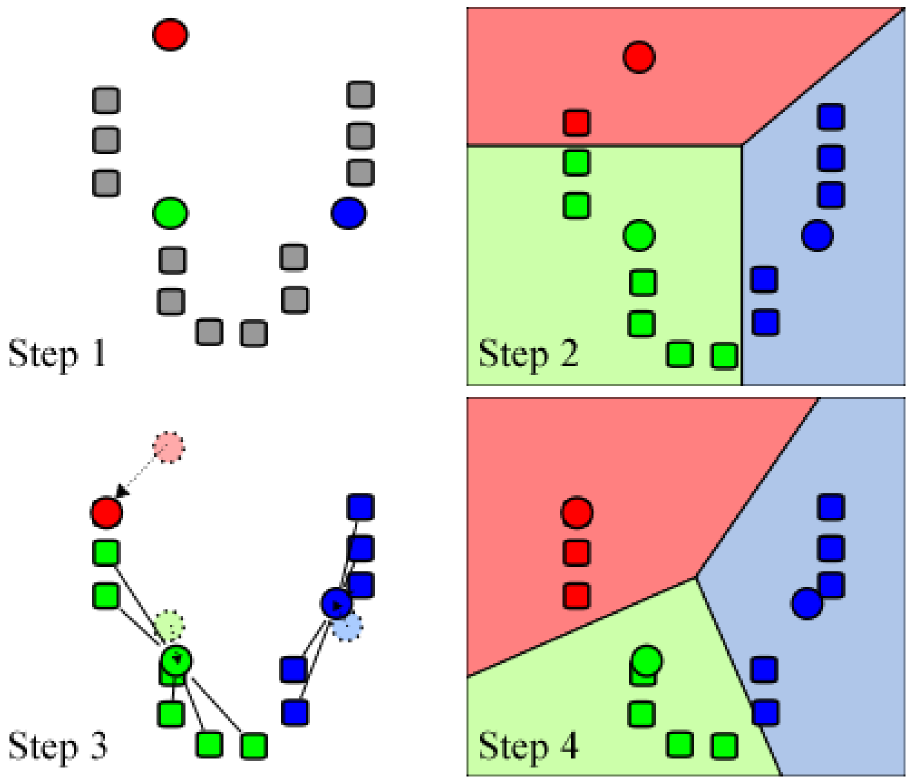


Figure 4.4: An illustration of the k-means algorithm for classifying objects.

mean of the objects that are assigned to this centroid. The second and third steps are repeated until convergence has been reached. In this illustration, the final three groups are presented in step 4.

We adapt the k-means algorithm with the aim of making it suitable for identifying neighbouring queries in the LTS framework (the 3rd step of Algorithm 4):

1. Set the parameter k, which is the number of groups.
2. Randomly or heuristically generate k centroids, which are in the range of a given query feature space.
3. Assign each training query to its nearest centroid, according to the query feature score (e.g., the KL divergence score). The queries that are assigned to the same centroid form a group.
4. For each group, update its centroid with the mean of its corresponding query feature scores.

5. Repeat the previous two steps until no training query moves from one group to another.

6. For a given test query q', find its nearest centroid.

7. The training queries that belong to the identified centroid are used in Equation (4.2) to decide which ranking function is the most appropriate for the test query q'.

4.4.2.3 The Bin Approach

Peng & Ounis (2009) proposed the use of bins to find an interval that a given query belongs to. Each interval corresponds to a sub-bin. The intervals are defined based on the feature scores of training queries. The algorithm steps are given as follows:

1. Set the parameter k, which is the number of sub-bins.

2. Divide the bin, which contains all training queries, into k equal size sub-bins according to the logscale of the feature scores of these training queries.

3. For a given test query q', find the sub-bin (interval) that it belongs to, according to its query feature score.

4. The training queries that belong to the identified sub-bin are used in Equation 4.2 to decide which ranking function is the most appropriate one for the test query q'.

By comparing the algorithms of the bin approach and k-means, we note that the bin approach is similar to k-means. Both approaches aim to generate k centroids/sub-bins based on the feature scores of training queries, and assign the test query to the nearest centroid or the sub-bin that it belongs to. The only difference is that k-means conducts several iterations to generate centroids while the bin approach only conducts one iteration to build sub-bins. In the learning to select framework, we employ KNN and k-means but not the bin approach, since it can be considered as a variant of k-means.

	MAP		r_1	r_2
q_ϕ	$E(q_\phi, r_1)$	$E(q_\phi, r_2)$	$f_1(r_1, q_\phi)$	$f_1(r_2, q_\phi)$
q_1	0.1	0.2	3	2
q_2	0.5	0.3	5	7
q_3	0.3	0.2	8	10
q_4	0.4	0.5	7	6
q_5	0.2	0.1	6	1
q_6	0.3	0.4	10	5
q_7	0.7	0.5	4	11
q_8	0.1	0.3	2	13

Table 4.1: An example of query feature scores and MAP evaluations for 10 training queries and 2 ranking functions.

4.4.3 Variants of Learning to Select

Given the components of the learning to select framework described above, several variants of this framework can be generated by suitably combining different versions of the components. When using the KNN algorithm to identify neighbouring queries, three variants are generated according to the used query feature: the mean of the relevance scores (Rel), the KL divergence score (KL) and the JS divergence score (JS). In the remaining chapters, we denote them as KNN-Rel, KNN-KL and KNN-JS respectively. In addition, another three variants can be generated when using k-means as the neighbouring query finding algorithm, the three variants are denoted as Kmeans-Rel, Kmeans-KL and Kmeans-JS.

4.5 Example of Learning to Select

Let us illustrate the learning to select framework using an example. Assuming our training dataset has 8 queries, namely $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$, and that we have two candidate ranking functions, namely $R = \{r_1, r_2\}$. The retrieval performance (e.g., MAP) of each ranking function and its query feature score for each training query q_ϕ are presented in Table 4.1.

Then, for an unseen query q', we estimate the query feature scores for each ranking function. In this example, we assume $f(r_1, q') = 2$ and $f(r_2, q') = 5$. In

	$d(q_\phi, q')$	
q_ϕ	r_1	r_2
q_1	1	3
q_2	3	2
q_3	6	5
q_4	5	1
q_5	4	4
q_6	8	0
q_7	2	6
q_8	0	8

Table 4.2: The distance between each training query q_ϕ and the given test query q', which is an absolute value of the subtraction between the two queries' feature scores.

addition, we employ the KNN algorithm and set $k = 3$ in order to find the three nearest neighbouring queries.

Next, we compute the distance between the given test query q' and each training query q_ϕ for each ranking function, according to the query feature score. The distance results are presented in Table 4.2.

Then, we sort the distance scores (shown in Table 4.3) and collect the three nearest neighbouring queries for each ranking function. In this case, q_1, q_7, q_8 are collected for r_1 and q_2, q_4, q_6 are collected for r_2.

Finally, the effectiveness of each candidate ranking function on this test query q' is computed as follows:

$$\begin{aligned}\frac{\sum_{\phi=1}^{k} E(q_\phi, r_1)}{k} &= \frac{E(q_1, r_1) + E(q_7, r_1) + E(q_8, r_1)}{3} \\ &= \frac{0.1 + 0.7 + 0.1}{3} = 0.3\end{aligned} \tag{4.10}$$

$$\begin{aligned}\frac{\sum_{\phi=1}^{k} E(q_\phi, r_2)}{k} &= \frac{E(q_2, r_2) + E(q_4, r_2) + E(q_6, r_2)}{3} \\ &= \frac{0.3 + 0.5 + 0.4}{3} = 0.4\end{aligned} \tag{4.11}$$

In this case, for q', we apply r_2 as its mean retrieval performance for the nearest queries is higher than for r_1 ($0.4 > 0.3$).

	$d(q_\phi, q')$		ranking		selected?	
q_ϕ	r_1	r_2	r_1	r_2	r_1	r_2
q_1	1	3	2	4	Yes	No
q_2	3	2	4	3	No	Yes
q_3	6	5	7	6	No	No
q_4	5	1	6	2	No	Yes
q_5	4	4	5	5	No	No
q_6	8	0	8	1	No	Yes
q_7	2	6	3	7	Yes	No
q_8	0	8	1	8	Yes	No

Table 4.3: The ranking of training queries according to the computed query distance scores.

4.6 Discussion

The proposed learning to select framework is different from the previously proposed selective retrieval approaches (Chapter 3) in several aspects.

First, the query dependent ranking approach (Geng et al., 2008) (described in Section 3.3) tries to learn a ranking function for a given query, through the way of applying a single learning to rank algorithm to the neighbouring queries of the given query. However, in this approach, it is not clear how to obtain a ranking function when there is more than one learning to rank algorithm. In contrast, our proposed learning to select framework selects an appropriate ranking function from a number of candidate ranking functions, which can be created by using several different learning to rank techniques, for each given query.

Three different variants of the query dependent ranking algorithm were proposed, namely KNN Online (Section 3.3.1), KNN Offline-1 (Section 3.3.2) and KNN Offline-2 (Section 3.3.3), which have different time complexities. Compared to KNN Online, the learning to select framework is much more efficient as the weights of each document feature are learned beforehand. However, to generate a ranking function, KNN Online learns the document feature weights for each given query at retrieval time, which is time-consuming, and results in the KNN Online algorithm being impractical.

To alleviate the time complexity during the retrieval process, KNN Offline-1 and KNN Offline-2 were proposed. Instead of conducting a single online training for each given query, KNN Offline-1 and KNN Offline-2 learn a specific ranking function for each training query before the retrieval process. However, compared to KNN Online, this kind of process significantly increases the training overheads, as n times training hours are spent if there are n training queries. In particular, the KNN Offline-1 algorithm introduces additional computation overheads for finding the neighbours of a given query, in comparison with KNN Offline-2.

Second, the selective collection enrichment approach (Section 3.4) selectively applies an appropriate collection for enriching and expanding a given query according to the given query's predicted performance on the corresponding collection. The performance prediction mainly relies on the statistics of a given query in a corresponding collection rather than a given ranking function. Therefore, the selective collection enrichment approach may not be applicable to the selective application of a ranking function, as these collection statistics are invariant to changes in the ranking function. However, our proposed learning to select framework is not only able to selectively apply an appropriate ranking function, but is also capable of selectively choosing an appropriate collection for expanding a given query. This is achieved since a same set of documents receive different relevance scores, according to different expanded queries, which are obtained by applying query expansion on different collections. In this case, each collection is represented by a list of document relevance scores.

Third, the selective Web IR framework (Section 3.5) is formulated in terms of statistical decision theory and based on a Bayesian decision mechanism. For a given query, the retrieval approach with the minimum expected loss is applied. However, the loss function employed in selective Web IR is loosely related with the target evaluation measure (e.g., MAP). Several studies have shown that it is advantageous to define the loss function directly in line with the target evaluation measure (Cossock & Zhang, 2006; Donald et al., 2005; Järvelin & Kekäläinen, 2000). In the learning to select framework, the decision mechanism is directly related with the target evaluation measure. For a given query, the ranking function that receives the highest estimated retrieval performance (e.g., MAP) on its neighbouring queries is applied. In addition, the retrieval performance of the selective Web IR framework is highly dependent on the number of candidate retrieval approaches. The obtained retrieval performance by using the selective

Web IR framework decreased when more than two candidate retrieval approaches are used. This is because, in the selective Web IR framework, a higher number of candidate retrieval approaches require more queries for training the Bayesian decision mechanism. However, in the learning to select framework, the effectiveness of each candidate ranking function is estimated based on its retrieval performance on a set of neighbouring training queries, which is independent of the size of candidate ranking functions.

Fourth, the query type prediction method (Section 3.6) attempts to identify the type of a given query, then applies different retrieval approaches for different query types. Instead, the learning to select framework selectively applies an appropriate ranking function on a per-query basis, which is agnostic to the type of the queries. The learning to select framework could bring more benefit to the retrieval performance as queries of the same type may benefit from having different retrieval approaches applied.

4.7 Summary

In this chapter, we presented a novel framework, called learning to select, for selecting an appropriate ranking function (see Section 4.2) or multiple document features (see Section 4.3). The framework is based on the concept that the effectiveness of a ranking function for an unseen query can be estimated based on its performance on similar already seen queries. The framework consists of two main components: computing a set of query features (Section 4.4.1) and identifying neighbouring queries (Section 4.4.2). In the learning to select framework, a classification algorithm is employed to identify neighbouring queries by using a query feature. Moreover, to build this query feature, we propose the use of a divergence measure to determine the extent to which a document ranking function alters the scores of an initial ranking of documents.

In Section 4.6, we discussed how the learning to select framework differs from the existing selective retrieval approaches, including query dependent ranking (Section 3.3), selective collection enrichment (Section 3.4), selective Web IR (Section 3.5), and query type prediction (Section 3.6). The remainder of this thesis focuses on evaluating the effectiveness of the learning to select framework in different search applications.

Chapter 5

Experiments using Learning to Select

5.1 Introduction

The previous chapter presented a novel learning to select (LTS) framework for selectively applying an appropriate ranking function and multiple document features for a given query. In this chapter, we aim to evaluate the performance of our proposed LTS framework on several different aspects.

Section 5.2 presents the research questions of this chapter. As described in Chapter 4, our proposed LTS framework is capable of selecting an appropriate ranking function and multiple appropriate document features. We investigate the effectiveness of our proposed LTS framework in both cases. In particular, the robustness of our proposed LTS framework is investigated by increasing the size of the candidate set. Furthermore, the performance of each component (i.e., query feature and the neighbouring query finding technique) of the LTS framework is also investigated.

Our experimental settings are described in Section 5.3. Section 5.4 shows the importance of selectively applying an appropriate ranking function on a per-query basis. Sections 5.5 and 5.6 discuss the evaluation of our proposed LTS framework when selecting a single and multiple candidates, respectively. We summarise this chapter and draw our conclusions in Section 5.7.

5.2 Research Questions

In the following experiments, we address five main research questions:

- RQ1: We assess how important it is to selectively apply an appropriate ranking function on a per-query basis (Section 5.4).
- RQ2: We test how effective our proposed LTS framework is for selecting an appropriate ranking function for a given query, by comparing it to three state-of-the-art LTR techniques and a simulated oracle query type prediction approach which knows with certainty the query type before applying a given ranking function (Section 5.5.1).
- RQ3: As the number of candidate ranking functions increases, the selection becomes more challenging. To test how robust our proposed LTS framework is, we apply it on a large number of candidate ranking functions, which contains as many as 63 different ranking functions (Section 5.5.2).
- RQ4: We test how effective our proposed LTS framework is when select multiple appropriate candidates (i.e. document features) from the candidate set (Section 5.6).
- RQ5: We test how effective each LTS component is, by investigating three different query features (i.e., the mean of the relevance scores, KL divergence, and JS divergence) and two different neighbouring query finding techniques (i.e., KNN and *k*-means) (Sections 5.5 and 5.6).

5.3 Experimental Settings

We present the experimental settings in this section, which includes the retrieval tasks (Section 5.3.1), the used collections and topics (Section 5.3.2), the used baselines (Section 5.3.3), and the training procedure (Section 5.3.4).

Number of pages	1,247,753
Average page size	15.2 kB
Number of hostnames	7,794
Total number of links	11,164,829
Number of cross-host links	2,470,109
Average cross-host links per host	317

Table 5.1: Salient properties of the .GOV corpus

5.3.1 Retrieval Tasks

The LETOR 3.0 and LETOR 4.0 feature sets (Liu et al., 2007) are sampled using datasets from two tracks: Web track and Million Query track. Web track 2003 and 2004 aim to formulate Web-specific search tasks (i.e., topic distillation task, named page finding task and homepage finding task), which are representative of common Web search activities (Craswell & Hawking, 2002). As we mentioned in Section 2.7, most relevance assessments in the context of the Text REtrieval Conference (TREC) are conducted on the merged document pool, which is developed based on the idea from Spärck Jones & van Rijsbergen (1976). For each query, the top K returned documents (normally $K = 100$) from a set of participating IR systems are merged into a document pool (Voorhees & Harman, 2000). The Million Query track 2007 and 2008 are proposed with the aim to investigate whether it is better to evaluate retrieval systems using many shallow judgments or instead fewer thorough judgments. Moreover, the Million Query track is an exploration of ad hoc retrieval on a large collection of documents (Allan et al., 2007).

5.3.2 Collections and Topics

The TREC Web track 2003 and 2004 are conducted on the .GOV corpus, which is a crawl of Web sites in the .gov domain from early 2002. The crawl includes binary and text mime types, and is stopped after 1 million HTML pages. The HTML text, plus the extracted text from other document types, amount to a total of 1.25 million documents (Craswell & Hawking, 2002). Some properties of .GOV are listed in Table 5.1. The Million Query track 2007 and 2008 are conducted on the .GOV2 corpus, which is crawled from Web sites in the .gov

Web track		
Topic Type	TREC 2003	TREC 2004
Homepage finding	150	75
Named Page finding	150	75
Topic Distillation	50	75
Total	350	225

Million Query track		
	TREC 2007	TREC 2008
Total	1692	784

Table 5.2: The number of queries for the TREC 2003 & 2004 Web Track and TREC 2007 & 2008 Million Query Track.

domain during early 2004 (Clarke et al., 2004). There are 25 million documents contained in the .GOV2 corpus, including HTML documents, plus the extracted text of PDF, Word and postscript files.

We use the TREC 2003 and 2004 Web track query sets, which contain three different topic types, namely homepage finding, named page finding and topic distillation. The target of the homepage finding task is to find the homepage of a site for a given query, while the target of the named page finding task is to find non-homepage pages for a given query. The topic distillation task involves finding homepages of relevant sites, given a broad query (Craswell & Hawking, 2002). The number of queries for each topic type can be found in Table 5.2. In search applications, users do not specify the type of their submitted query. In order to simulate a real IR environment, we mix the three topic types together. The query sets of the TREC 2007 and 2008 Million Query tracks are drawn from a large collection of queries that are collected by a large Internet search engine. The number of assessed queries from the TREC 2007 and TREC 2008 Million Query tracks can also be found in Table 5.2.

Two feature sets are used in our experiments, namely LETOR 3.0 and LETOR 4.0 (Liu et al., 2007). The LETOR 3.0 dataset contains 64 different document features, including document length and HostRank (Xue et al., 2005), among others. The documents in LETOR 3.0 are sampled from the top retrieved documents by using BM25 (Equation (2.6)) on the .GOV corpus with the TREC 2003 and TREC 2004 Web track queries. In contrast, the LETOR 4.0 dataset contains

46 document features for documents similarly sampled from the .GOV2 corpus using the TREC 2007 and TREC 2008 Million Query track queries. The detailed features included in LETOR 3.0 and LETOR 4.0 are presented in Table A.1 and Table A.2, respectively (see appendix).

5.3.3 Retrieval Baselines

Many different learning to rank techniques have been proposed during the past few years. Due to the fact that the pointwise learning to rank approaches treat documents separately, the relative order between documents is invisible to the loss functions. To build a strong baseline, in this work, we use the pairwise and listwise learning to select approaches. In particular, three state-of-the-art learning to rank techniques are employed: Ranking SVM (Herbrich et al., 2000; Joachims, 2002), AdaRank (Xu & Li, 2007) and the AFS method (Metzler, 2007). These learning to rank methods have been described in detail in Section 3.2.

All of these learning to rank techniques are used as the retrieval baseline, in comparison with our proposed learning to select framework. Moreover, for the Web track, we simulate an oracle Query Type Prediction (QTP) approach as another baseline. The simulated query type prediction approach knows with certainty the query type before applying a ranking function. In addition, the best performing ranking function for each query type, which can be learned on the training dataset, is applied for an identified query type.

5.3.4 Training Procedure

BM25 is used as our base ranking function as the features included in the LETOR datasets are computed over the top retrieved documents, which are sampled using BM25 (Liu et al., 2007). The feature weights that are related with each candidate ranking function by using the AFS approach are set by optimising Mean Average Precision (MAP) on the training dataset, using a simulated annealing procedure (Skiścim & Golden, 1983). The number of the top ranked documents (T) and the number of neighbouring queries (K), introduced in Chapter 4, are also set by optimising MAP over the training dataset, using a large range of dif-

	TREC 2003				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.5367	0.1811	0.1043	0.6396	0.6528
AdaRank	0.6038	0.1754	0.0997	0.6663	0.6785
AFS	0.6117	0.1766	0.1051	0.6748	0.6959
Upper Bound	**0.6915** $\star \ast \dagger$	**0.2046** $\star \ast \dagger$	**0.1180** $\star \ast \dagger$	**0.7590** $\star \ast \dagger$	**0.7725** $\star \ast \dagger$

	TREC 2004				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.4185	0.2098	0.1404	0.5398	0.5530
AdaRank	0.4901	0.2044	0.1333	0.5844	0.5986
AFS	0.4943	0.2196	0.1418	0.6062	0.6127
Upper Bound	**0.5700** $\star \ast \dagger$	**0.2560** $\star \ast \dagger$	**0.1622** $\star \ast \dagger$	**0.6831** $\star \ast \dagger$	**0.6758** $\star \ast \dagger$

	TREC 2007				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.4615	0.4082	0.3820	0.4076	0.4391
AdaRank	0.4597	0.4025	0.3749	0.4056	0.4353
AFS	0.4591	0.4068	0.3750	0.4128	0.4381
Upper Bound	**0.5016** $\star \ast \dagger$	**0.4643** $\star \ast \dagger$	**0.4171** $\star \ast \dagger$	**0.4782** $\star \ast \dagger$	**0.4964** $\star \ast \dagger$

	TREC 2008				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.4714	0.3434	0.2489	0.4671	0.2284
AdaRank	0.4735	0.3446	0.2478	0.4723	0.2297
AFS	0.4766	0.3487	0.2474	0.4782	0.2304
Upper Bound	**0.5146** $\star \ast \dagger$	**0.3708** $\star \ast \dagger$	**0.2578** $\star \ast \dagger$	**0.5168** $\star \ast \dagger$	**0.2509** $\star \ast \dagger$

Table 5.3: For each dataset, the highest score in each column is highlighted in bold and scores that are statistically better than $RankingSVM$, $AdaRank$, and AFS are marked with $\star$, $\ast$, and $\dagger$, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$).

ferent value settings. We evaluate our experimental results using MAP, Precision at N, and normalised Discounted Cumulative Gain (nDCG).

In our experiments, we divide each of the four datasets (Table 5.2) into five equal size folds. We iteratively test our LTS framework on one fold by using another fold as a validation set and the remaining three folds as a training set. We report the obtained results and their analysis in the following section.

5.4 The Importance of Selective Application

In order to assess how important it is to selectively apply an appropriate ranking function. We produce the upper bounds of the retrieval performance by manually selecting the most effective ranking function on a per-query basis.

Table 5.3 shows the retrieval performance of the three learning to rank techniques (i.e. Ranking SVM, AdaRank and the AFS method) and Upper Bound (100% correct per-query application) on four different datasets. From this table, it is clear that the retrieval performance can be significantly enhanced if we apply the most appropriate ranking function for each query. This observation suggests that different ranking functions do favour different queries and that the appropriate selective application of a ranking function could significantly enhance the retrieval performance.

5.5 Selecting A Single Ranking Function

In this section, we investigate how effective our proposed learning to select framework is for selecting an appropriate ranking function on a per-query basis (Section 5.5.1). In addition, as the number of candidate ranking functions increases, the selection becomes more challenging. To test how robust our proposed learning to select framework is, we apply it on a large number of candidate ranking functions (Section 5.5.2).

5.5.1 The Effectiveness of Learning to Select

In order to test the effectiveness of our proposed framework for selectively applying an appropriate ranking function for a given query, we compare it with the simulated oracle query type prediction approach, and the three state-of-the-art learning to rank techniques, which are systematically applied to all queries. In addition, six variants that are derived from the learning to select framework are investigated. KNN-rel, KNN-KL and KNN-JS denote that we employ the KNN algorithm for identifying similar queries based on the query feature of the mean of the relevance score, the KL divergence score and the JS divergence score,

	TREC 2003				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.5367 †	<u>0.1811</u>	0.1043	0.6396 †	0.6528 †
AdaRank	0.6038 †	0.1754	0.0997	0.6663 †	0.6785 †
AFS	<u>0.6117</u> †	0.1766	**0.1051**	<u>0.6748</u> †	<u>0.6959</u> †
QTP	0.6126 †	0.1789	0.1031	0.6801	0.6949
Learning to Select					
KNN-Rel	0.6271 $*$ $\diamond$	0.1800	0.1040	0.6793	0.6926
KNN-KL	0.6362 $*$ $\diamond$	0.1811	0.1034	**0.6950** $*$ $\diamond$	0.6997
KNN-JS	0.6364 $*$ $\diamond$	0.1806	0.1046	0.6915 $*$	0.6993
Kmeans-Rel	0.6219 $*$	**0.1834**	**0.1051**	0.6815	0.6935
Kmeans-KL	**0.6425** $*$ $\diamond$	0.1783	0.1034	0.6921 $*$	0.7049
Kmeans-JS	0.6418 $*$ $\diamond$	0.1777	0.1026	0.6880 $*$	**0.7090** $*$ $\diamond$

Table 5.4: Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 3.0 TREC 2004 dataset. Results are the mean over 5 folds.

respectively; Kmeans-rel, Kmeans-KL and Kmeans-JS denote that we use the k-means algorithm for identifying similar queries based on the query feature of the mean of the relevance score, the KL divergence score and the JS divergence score, respectively.

Tables 5.4, 5.5, 5.6 and 5.7 present the evaluation of the retrieval performances obtained by using the three state-of-the-art LTR techniques, the oracle query type prediction (QTP) approach and by applying our proposed LTS framework in terms of MAP, Precision at N & nDCG on the LETOR 3.0 & LETOR 4.0 datasets, respectively. The best retrieval performances for each evaluation measure on each different dataset are emphasised in bold. The † symbol indicates that the retrieval performance obtained by the best variant of our proposed LTS framework is significantly better than the compared baseline, according to the Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$). The best retrieval performance obtained by systematically applying an LTR technique on all queries is highlighted with underline. The $*$ and $\diamond$ symbols indicate that the retrieval performance obtained by using our proposed LTS framework is significantly better than the underlined score and the QTP approach, respectively.

	TREC 2004				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.4185 †	0.2098	0.1404	0.5398 †	0.5530 †
AdaRank	0.4901 †	0.2044	0.1333	0.5844 †	0.5986 †
AFS	<u>0.4943</u> †	**<u>0.2196</u>**	**<u>0.1418</u>**	<u>0.6062</u>	<u>0.6127</u>
QTP	0.4976 †	0.2124	0.1364	0.5963	0.6092
Learning to Select					
KNN-Rel	0.4998	0.2133	0.1356	0.5800	0.5971
KNN-KL	0.5229 $*$ $\diamond$	0.2062	0.1347	0.6124 $\diamond$	0.6129
KNN-JS	0.5243 $*$ $\diamond$	0.2116	0.1360	**0.6138** $\diamond$	0.6134
Kmeans-Rel	0.5047	0.2080	0.1369	0.6074	0.6011
Kmeans-KL	0.5216 $*$ $\diamond$	0.2133	0.1364	0.6039	**0.6137**
Kmeans-JS	**0.5259** $*$ $\diamond$	0.2098	0.1382	0.6026	0.6115

Table 5.5: Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 3.0 TREC 2004 dataset. Results are the mean over 5 folds.

From the results in Tables 5.4, 5.5, 5.6 and 5.7, we observe that, in general, the best retrieval performance in each column is achieved by using our proposed LTS framework, e.g., on the LETOR 3.0 TREC 2003 dataset (Table 5.4), the highest score in MAP is 0.6425. The only exceptions are for the *Precision* evaluation measure. However, in each exception case, the performance of LTS is close to the highest *Precision* score. For example, the best retrieval performance in P@5 on the LETOR 4.0 TREC 2008 dataset (Table 5.7) is 0.3487 and the retrieval performance obtained by using our LTS framework is 0.3482 (Kmeans-KL).

Furthermore, from the MAP column, the best retrieval performance obtained by using our proposed LTS framework constantly outperforms all the LTR techniques across all datasets, e.g., on the TREC 2003 dataset (Table 5.4): 0.5367 $\rightarrow$ 0.6425; 0.6038 $\rightarrow$ 0.6425; and 0.6117 $\rightarrow$ 0.6425. In particular, these improvements are statistically significant. Moreover, compared to the oracle QTP approach that is obtained by simulating an ideal 100% accuracy in detecting the query type, we can see that our proposed method constantly outperforms the QTP method on both the TREC 2003 and TREC 2004 datasets. For example, in the MAP measure, 0.6126 $\rightarrow$ 0.6425 on the TREC 2003 dataset (Table 5.4); and 0.4976

	TREC 2007				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	<u>0.4615</u> †	<u>0.4082</u>	<u>**0.3820**</u>	0.4076	<u>0.4391</u>
AdaRank	0.4597 †	0.4025 †	0.3749	0.4056 †	0.4353 †
AFS	0.4591 †	0.4068	0.3750	<u>0.4128</u>	0.4381
Learning to Select					
KNN-Rel	0.4654	0.4078	0.3760	0.4110	0.4380
KNN-KL	0.4696	**0.4146**	0.3790	0.4157	0.4440
KNN-JS	0.4681	0.4080	0.3779	**0.4172**	**0.4443**
Kmeans-Rel	0.4632	0.4087	0.3819	0.4107	0.4403
Kmeans-KL	**0.4707** ∗	0.4123	0.3810	0.4162	0.4433
Kmeans-JS	0.4698 ∗	0.4115	0.3816	0.4155	0.4436

Table 5.6: Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 4.0 TREC 2007 dataset. Results are the mean over 5 folds.

$\rightarrow$ 0.5259 on the TREC 2004 dataset (Table 5.5). This particularly stresses the effectiveness of our approach, given that the query type prediction in a practical system is usually much lower than 100%. It also suggests that queries which have the same type do not necessarily equally benefit from the application of a given ranking function.

From the results in tables 5.4, 5.5, 5.6 & 5.7, we observe that the JS divergence measure and the KL divergence measure are producing very close retrieval performances and both of them consistently outperform the mean of the relevance scores. For example, by using the KNN algorithm on the TREC 2003 dataset: 0.6271 vs. 0.6362; and 0.6271 vs. 0.6364. This is explained by the fact that they are mathematically related. Particularly, in some cases, the retrieval performances obtained by using the divergence measures (i.e. KL and JS) make significant improvement over the baselines (i.e., the best LTR technique and the QTP baseline) while the mean of the relevance scores does not. For example, on the TREC 2004 dataset (Table 5.5), KNN-KL and KNN-JS make significant improvements over AFS (0.5229 vs. 0.4943 and 0.5243 vs. 0.4943) while KNN-Rel does not (0.4998 vs. 0.4943). By comparing the two different neighbouring query finding techniques, we observe that *k*-means is slightly better

	TREC 2008				
	MAP	P@5	P@10	nDCG@5	nDCG@10
Ranking SVM	0.4714 †	0.3434	<u>0.2489</u>	0.4671 †	0.2284
AdaRank	0.4735 †	0.3446	0.2478	0.4723	0.2297
AFS	<u>0.4766</u> †	**0.3487**	0.2474	<u>0.4782</u>	<u>0.2304</u>
Learning to Select					
KNN-Rel	0.4806	0.3477	0.2471	0.4730	0.2284
KNN-KL	0.4859 ∗	0.3464	0.2473	0.4731	0.2286
KNN-JS	0.4857 ∗	0.3477	0.2487	0.4732	**0.2322**
Kmeans-Rel	0.4805	0.3477	**0.2491**	0.4727	0.2304
Kmeans-KL	**0.4860** ∗	0.3482	**0.2491**	**0.4794**	0.2303
Kmeans-JS	0.4843 ∗	0.3451	0.2477	0.4769	0.2314

Table 5.7: Comparison between LTS and state-of-the-art LTR techniques using different evaluation measures on the LETOR 4.0 TREC 2008 dataset. Results are the mean over 5 folds.

than KNN and always produces the highest retrieval performance in MAP, e.g., on the TREC 2003 dataset, the best retrieval performance is 0.6425, which is obtained by Kmeans-KL.

The above observations suggest that our proposed LTS framework is effective in applying an appropriate ranking function on a per-query basis. In addition, both KNN and k-means are effective in finding neighbouring queries. Moreover, our proposed use of divergence measures as a query feature is more effective than the use of the mean of the relevance scores.

5.5.2 The Robustness of Learning to Select

The analysis in Section 5.5.1 demonstrates the effectiveness of the proposed LTS framework on a small candidate set, which contains three different ranking functions that are learned by using three different learning to rank techniques. In this section, we investigate the robustness of our LTS framework when the number of candidate ranking functions increases. To achieve this, we simulate a number of candidate ranking functions by applying a single LTR technique on several different combinations of document features. In particular, in order to have a strong baseline, we use AFS, since it produces higher retrieval performances on average than the other two LTR techniques (e.g. see Tables 5.4 and 5.5).

In this investigation, six different document features are chosen from the LETOR feature set. There are then $2^6 - 1$ possible combinations (excluding the empty combination). Integrating each combination into the base ranking function (namely BM25) by using AFS produces one candidate ranking function. In this case, our task becomes to select an appropriate ranking function from a set of candidate ranking functions, which contains as many as $2^6 - 1 = 63$ candidate ranking functions.

Figures 5.1, 5.2, 5.3 and 5.4 show the effect on MAP as the number of candidate ranking functions is varied, on different TREC datasets. We order the 63 ranking functions according to their performance on the training dataset and the best performing ranking function is used as our baseline. The x axis denotes the number of top performing ranking functions applied. For example, for TREC 2003, 10 in the x axis means that we use the top 10 performing ranking functions, which are assessed on the training dataset, as candidate ranking functions to selectively apply on the test dataset.

From Figures 5.1, 5.2, 5.3 and 5.4, we observe that all variants derived from our LTS framework consistently outperform the baseline when the number of candidate ranking functions is increased. For the used query feature, the KL divergence and the JS divergence have similar distributions. In particular, the divergence measures (i.e. KL and JS) always perform better than the mean of the relevance scores (i.e. Rel), with both KNN and k-means. This is particularly noticeable for the TREC 2004 by using KNN (Figure 5.2(a)).

Indeed, in contrast to the *Rel* measure, both *JS* and *KL* are enhanced as the number of candidate ranking functions increases. For example, in the TREC 2003 KNN case (see Figure 5.1 (a)), the retrieval performances obtained by using the KL and JS measures in general increase from 2 to 35. This is mainly because each newly added ranking function has different behaviour and favours different queries. Hence, with more ranking functions added, the retrieval performance can be further improved if they can be selectively applied in an appropriate manner.

However, the retrieval performance only improves slightly when the number of candidate ranking functions keeps increasing after 35. The reason for this is that these ranking functions are created based on the combination of a small set of features, most of them with similar behaviour, which results in these newly added ranking functions (from 35 to 50 in the TREC 2003 KNN case) favouring the same queries as the previously chosen ranking functions.

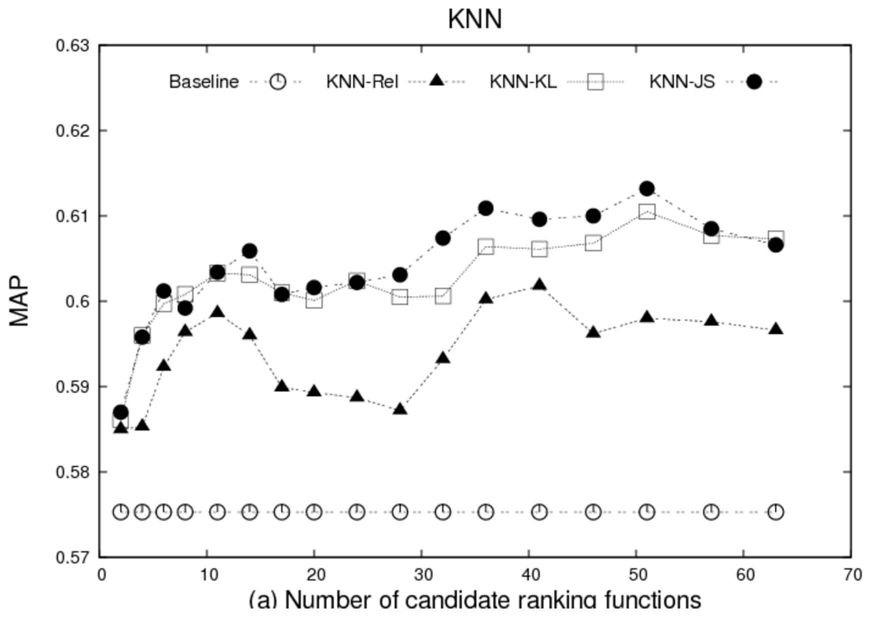
We also observe that the best retrieval performance is obtained when there are around 50 candidate ranking functions in the TREC 2003 KNN case (Figure 5.1(a)). However, the performance starts to decrease after 50 – as the last added ranking functions perform poorly on the training dataset and bring noise to the LTS framework. In addition, by comparing the retrieval performances obtained by the two different neighbouring query finding techniques in Figures 5.1, 5.2, 5.3 & 5.4, we observe that KNN and Kmeans have a similar distribution across all datasets.

This investigation suggests that our method is a robust approach, which can increasingly improve the retrieval performance as more ranking functions are added. The only caveat is when the most poorly-performing ranking functions are added to the candidate set. However, this can be controlled by setting the number of top-performing candidate ranking functions from which to select. In addition, both KNN and k-means are effective in finding neighbouring queries, and our proposed use of divergence measures as a query feature is more effective than the use of the mean of the relevance scores, which is in line with the observation in Section 5.5.1.

5.6 Selecting Multiple Document Features

Section 5.5 demonstrated the effectiveness and robustness of our proposed LTS framework for selecting a single ranking function from many candidate ranking functions, which are built on a fixed set of document features. However, in some cases, we need to selectively use multiple appropriate document features for building a ranking function for a given query.

In this section, we investigate the effectiveness of our learning to select framework for selecting multiple appropriate document features. In particular, the investigation is conducted with the number of candidate document features ranging from 2 to 6. There are then $2^2 - 1$, $2^3 - 1$, $2^4 - 1$, $2^5 - 1$ and $2^6 - 1$ possible



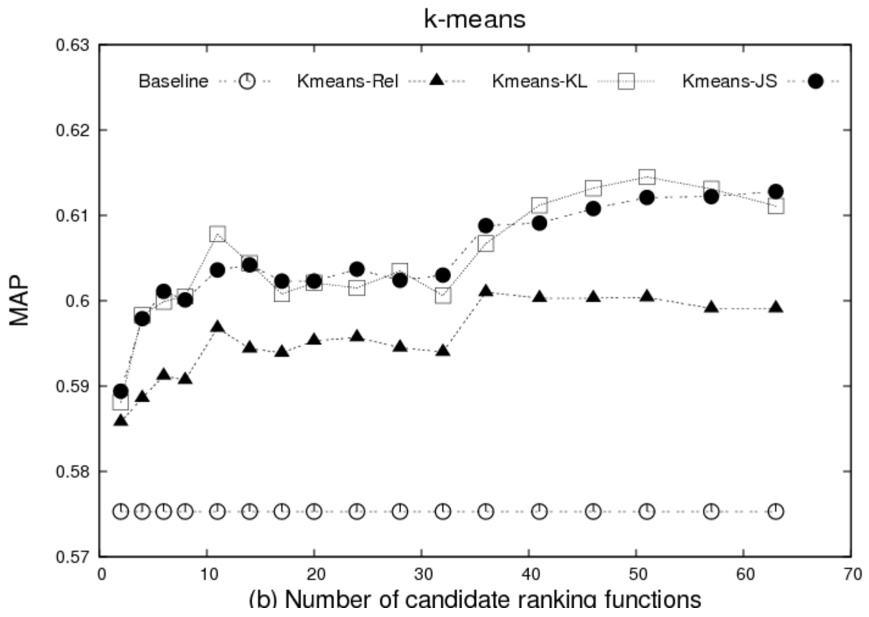
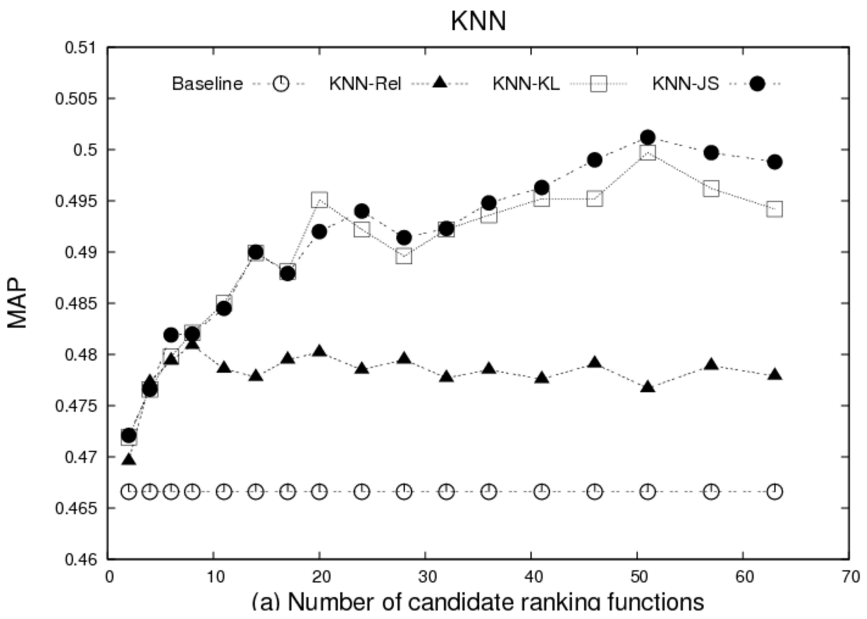


Figure 5.1: MAP versus the number of candidate ranking functions on the TREC 2003 dataset.



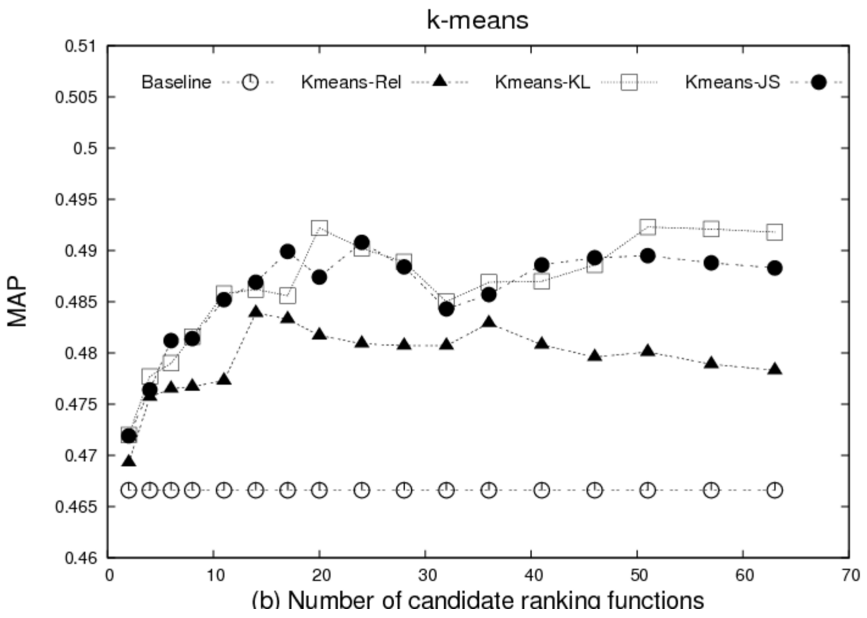
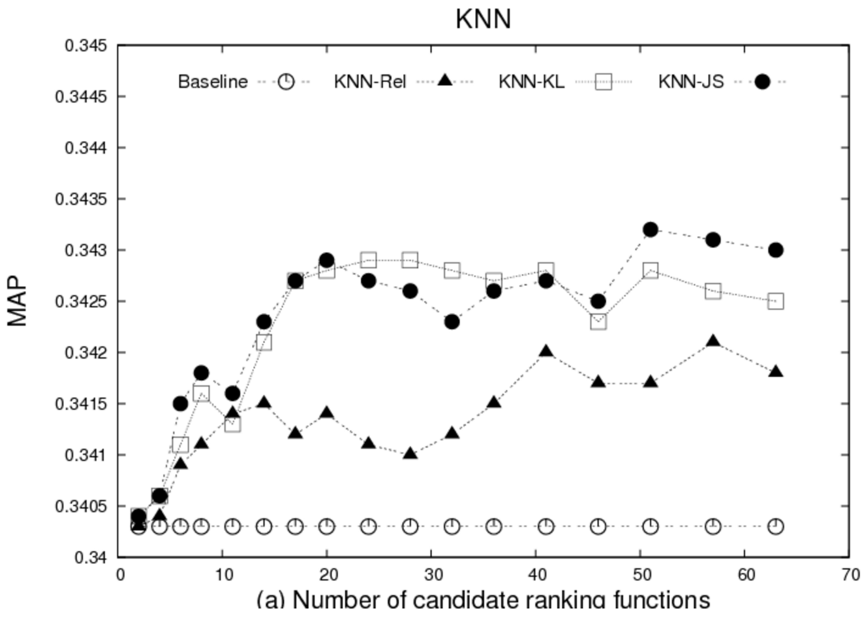


Figure 5.2: MAP versus the number of candidate ranking functions on the TREC 2004 dataset.



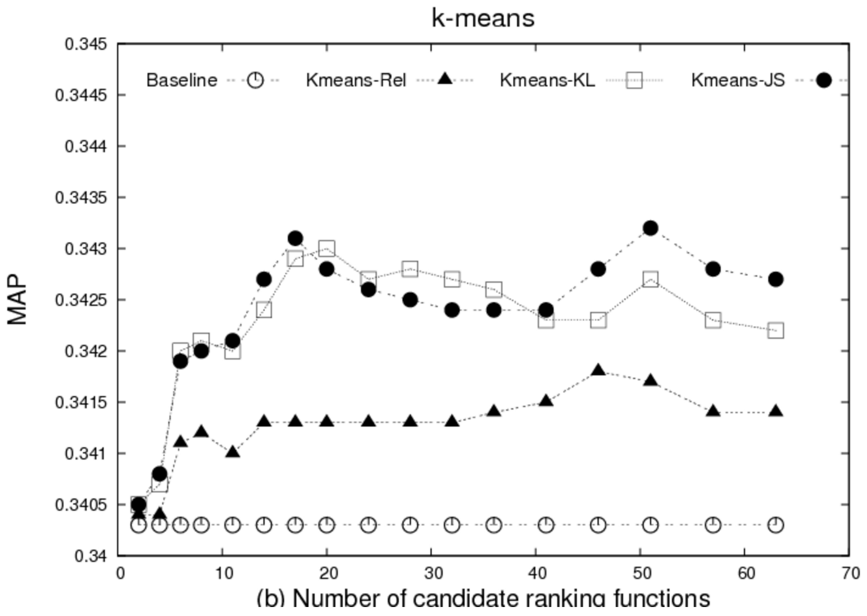
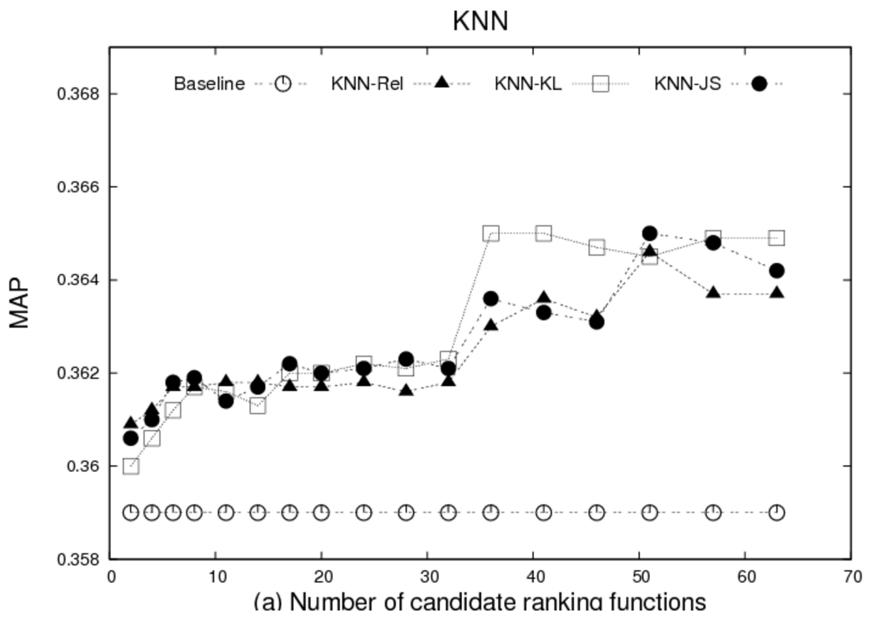


Figure 5.3: MAP versus the number of candidate ranking functions on the TREC 2007 dataset.



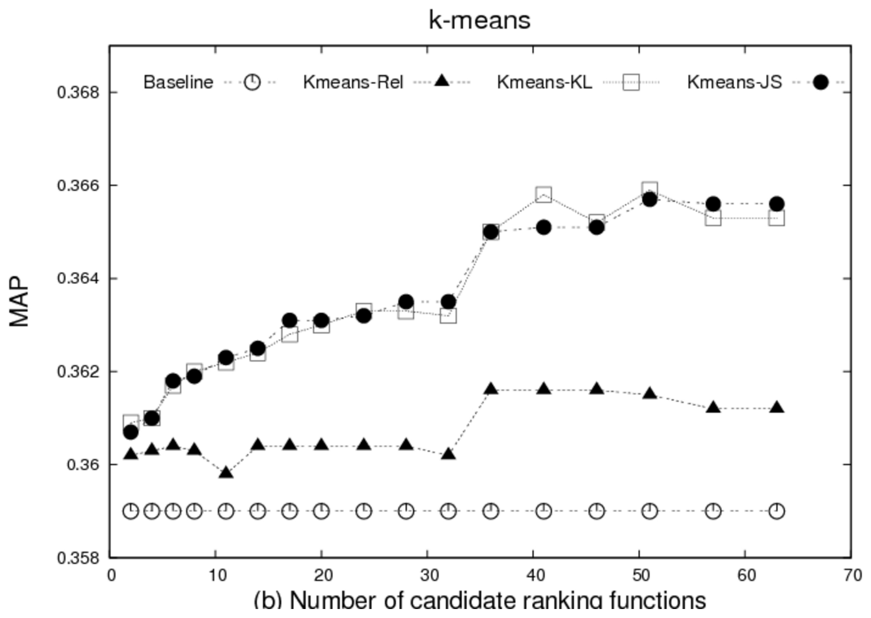


Figure 5.4: MAP versus the number of candidate ranking functions on the TREC 2008 dataset.

combinations (excluding the empty combination) for each case. In this investigation, a candidate ranking function is created by integrating a combination into a base ranking function (BM25 in this case). Moreover, in order to have a strong baseline, we use AFS to learn the weight of each document feature that included in each combination, since it produces on average higher retrieval performance than the other two LTR techniques (e.g. see Tables 5.4 and 5.5).

Tables 5.8, 5.9, 5.10 and 5.11 present the evaluation of the retrieval performances obtained by using different variants of the learning to select framework on different TREC datasets in terms of MAP. For each multiple selection, the systematic application of the best performing combination is used as our baseline. The best retrieval performances on different candidate sizes are emphasised in bold. The $*$ symbol indicates that the retrieval performance obtained by the variants of our proposed LTS framework is significantly better than the baseline, according to the Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$).

From Tables 5.8, 5.9, 5.10 and 5.11, we observe that the baseline performance is further enhanced when we increase the number of candidate document features from 2 to 6. For example, in Table 5.8, the baseline retrieval performance is increases from 0.5463 to 0.5823 when the number of candidate document features is increased from 2 to 6. This is probably because a larger number of effective document features used for ranking documents, a better retrieval performance can be obtained. This is also in line with the current growing trend which consists in applying an LTR technique on a large set of features in order to obtain a more effective ranking function.

We also observe that the retrieval performances obtained by the learning to select framework always make improvement over the baseline on various sizes of candidate document features and on all TREC datasets. For example, in Table 5.8, with two candidate document features, the retrieval performance is enhanced from 0.5463 (Baseline) to 0.5928 (Kmeans-JS). In particular, such improvements are significant in most cases on the TREC 2003 and 2004 datasets. Moreover, the best retrieval performance on each TREC dataset is obtained on the feature set that with the largest number of candidate document features (6 in our this investigation).

	MAP				
	2	3	4	5	6
Baseline	0.5463	0.5689	0.5694	0.5801	0.5823
KNN-Rel	0.5711 $*$	0.5813	0.5823	0.5881	0.5966
KNN-KL	0.5838 $*$	0.5918 $*$	0.5997 $*$	0.6042 $*$	0.6073 $*$
KNN-JS	0.5782 $*$	0.5895 $*$	0.5959 $*$	0.6105 $*$	0.6007 $*$
Kmeans-Rel	0.5692 $*$	0.5869 $*$	0.5875 $*$	0.5947	0.5991
Kmeans-KL	0.5888 $*$	0.5931 $*$	**0.6021** $*$	**0.6120** $*$	0.6111 $*$
Kmeans-JS	**0.5928** $*$	**0.5953** $*$	0.6004 $*$	0.6071 $*$	**0.6128** $*$

Table 5.8: The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2003 dataset with different numbers of candidate document features. Results are the mean over 5 folds.

In addition, we note that the close retrieval performances obtained by using *KL* and *JS*, e.g. in Table 5.8 and with six candidate document feature, 0.6111 (Kmeans-KL) vs. 0.6128 (Kmeans-JS). Moreover, both divergence measures consistently produce higher retrieval performances than the mean of the relevance scores (Rel). This suggests that our proposed use of divergence measures as a query feature is more effective than the mean of the relevance scores, which is in line with the observation found in Section 5.5. Furthermore, we observe that both KNN and k-means are producing similar retrieval performances, e.g. in Table 5.9 and by using KL as a query feature, 0.4691 (KNN-KL) vs. 0.4644 (Kmeans-KL).

The above observations suggest that our proposed learning to select framework is also effective for selecting multiple document features for building a ranking function on a per-query basis. In addition, the robustness of the learning to select framework is also observed as it is effective with various different sizes of candidate document features. Furthermore, the investigation on the components of the learning to select framework suggests that both KNN and Kmeans are effective in finding neighbouring queries and our proposed use of divergence measures as a query feature is more effective than the mean of the relevance scores, which is in line with the observation in Section 5.5.

	MAP				
	2	3	4	5	6
Baseline	0.4463	0.4644	0.4656	0.4676	0.4686
KNN-Rel	0.4495	0.4657	0.4767	0.4794	0.4779
KNN-KL	0.4691 ∗	0.4830 ∗	**0.4881** ∗	0.4876 ∗	0.4942 ∗
KNN-JS	**0.4692** ∗	0.4817 ∗	0.4858 ∗	**0.4951** ∗	**0.4976** ∗
Kmeans-Rel	0.4528	0.4695	0.4793	0.4796	0.4801
Kmeans-KL	0.4644 ∗	**0.4844** ∗	0.4858 ∗	0.4887 ∗	0.4923 ∗
Kmeans-JS	0.4670 ∗	0.4809 ∗	0.4850 ∗	0.4917 ∗	0.4891 ∗

Table 5.9: The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2004 dataset with different numbers of candidate document features. Results are the mean over 5 folds.

	MAP				
	2	3	4	5	6
Baseline	0.3386	0.3388	0.3398	0.3403	0.3403
KNN-Rel	0.3411	0.3412	0.3416	0.3425	0.3418
KNN-KL	0.3415	0.3414	0.3421	**0.3426**	0.3428
KNN-JS	**0.3417**	**0.3418**	**0.3426**	**0.3426**	**0.3432**
Kmeans-Rel	0.3403	0.3405	0.3410	0.3415	0.3414
Kmeans-KL	0.3412	0.3411	0.3412	0.3425	0.3422
Kmeans-JS	0.3413	0.3413	0.3417	**0.3426**	0.3427

Table 5.10: The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2007 dataset with different numbers of candidate document features. Results are the mean over 5 folds.

	MAP				
	2	3	4	5	6
Baseline	0.3614	0.3616	0.3618	0.3621	0.3625
KNN-Rel	0.3627	0.3635	0.3638	0.3639	0.3637
KNN-KL	0.3630	0.3647	**0.3650**	0.3645	0.3649
KNN-JS	0.3632	0.3642	0.3643	0.3637	**0.3658**
Kmeans-Rel	0.3619	0.3630	0.3635	0.3633	0.3632
Kmeans-KL	**0.3637**	0.3647	0.3641	**0.3649**	0.3653
Kmeans-JS	0.3631	**0.3651**	0.3646	0.3648	0.3656

Table 5.11: The evaluation of the LTS framework for selectively applying multiple document features on the TREC 2008 dataset with different numbers of candidate document features. Results are the mean over 5 folds.

5.7 Summary

In this chapter, we tested our Learning to Select (LTS) framework on the LETOR 3.0 & LETOR 4.0 feature sets and their corresponding TREC tasks. In particular, we compared it with a simulated query type prediction approach that with a 100% accuracy in predicting the type of a given query as well as three state-of-the-art Learning To Rank (LTR) techniques, namely Ranking SVM, AdaRank, and the AFS method.

Our experimental results showed that the retrieval performance obtained by using our proposed LTS framework could constantly outperform the query type prediction approach and three state-of-the-art techniques in the MAP measure on different datasets (see Section 5.5.1). In addition, improvements over the query type prediction approach and all LTR techniques were statistically significant in most cases.

Moreover, we investigated the effectiveness of our framework when the number of candidate ranking functions increases. By plotting the distribution of MAP versus the number of candidate ranking functions, we found that by using our proposed framework, the retrieval performance was enhanced when increasing the number of candidate ranking functions (see Section 5.5.2). This suggests that learning to select is a robust framework for selectively applying an appropriate ranking function.

In addition, we tested the effectiveness of our framework for selectively applying multiple appropriate document features for building a ranking function on a per-query basis. Experimental results showed that our proposed framework is effective for selecting multiple appropriate document features and the obtained retrieval performance can be further enhanced when we increase the size of the candidate document features.

Furthermore, our proposed use of divergence measures as query features to identify neighbouring queries was always more effective than the mean of the relevance scores measure, which ignores the distribution of relevance scores. Besides, both KNN and Kmeans are effective approach for finding neighbouring queries for information retrieval.

Chapter 6

Learning to Select in Other Search Applications

6.1 Introduction

The previous chapter evaluated the performance of the learning to select framework on two different tasks: selecting an appropriate ranking function from a number of candidate ranking functions and selecting multiple document features from a number of candidate document features for building a ranking function. In this chapter, we show how the learning to select framework can be deployed in other search applications.

In Section 6.2, we experiment to determine if the learning to select framework can be effectively applied to choose an appropriate query independent feature. This allows different document features to be applied to the queries that are more likely to benefit from these features. Moreover, we test how effective the learning to select framework is, by applying it to the xQuAD search diversification framework, in which an appropriate sub query importance estimator is selectively applied (Section 6.3). This allows the ranked list of documents to provide a complete coverage for an ambiguous query. Lastly, we examine the effectiveness of our proposed learning to select framework in choosing an appropriate resource for expanding an initial query (Section 6.4). This allows to alleviate the problem of mismatch between query terms and intranet documents, which are usually with a limited use of lexical representations. Section 6.5 draws the conclusions based on the observations found in Sections 6.2, 6.3 and 6.4.

6.2 Selective Application of Query Independent Features

The experiments conducted in Chapter 5 were based on the use of a learning to rank technique on a large number of document features, with the aim of building an effective ranking function for retrieval. However, several previous researchers (Craswell et al., 2005; Kamps et al., 2004; Kraaij et al., 2002; Lioma et al., 2006; Macdonald et al., 2008; Metzler et al., 2005; Peng et al., 2007) focused on the use of a small number of selected query independent document features, and showed that the retrieval performance can be boosted if an appropriate query independent document feature is used. In particular, the FLOE method, which transforms a query independent document feature value into a document relevance score (introduced in Section 2.6.4), has shown its effectiveness in the work of various authors (Craswell et al., 2005; Hannah et al., 2007; Serdyukov et al., 2008). However, most of this research systematically applies the obtained relevance scores to all queries, which ignores the fact that different queries benefit differently from different query independent document features.

In this section, we deploy the learning to select framework for selectively integrating an appropriate query independent document feature into a document weighting scheme, on a per-query basis. In particular, in the learning to select framework, the retrieval strategy without the use of query independent features is set as the base ranking function (see Section 4.4.1), and the retrieval strategies that use a given query independent feature, are set as the candidate ranking functions (see Section 4.4.1). For the remainder of this section, we introduce our research questions in Section 6.2.1; the settings for our experiments in Section 6.2.2; and the experimental results and analysis in Section 6.2.3. Conclusions of the selective application of a query independent feature are drawn in Section 6.2.4.

6.2.1 Research Questions

In the following experiments, we address three main research questions:

	TREC 2003			TREC 2004		
	HP	NP	TD	HP	NP	TD
Number of topics	150	150	50	75	75	75
Percentage	42.9%	42.9%	14.2%	33.3%	33.3%	33.3%

Table 6.1: Details of the number and percentage of topics associated to each topic type for TREC 2003 and TREC 2004 Web tracks.

- RQ1: we assess how important it is to selectively apply an appropriate query independent feature on a per-query basis (Section 6.2.3.1).
- RQ2: we test how effective the learning to select framework is at selectively applying one appropriate query independent feature out of two candidate features (Section 6.2.3.2).
- RQ3: as the number of candidate features increases, the selective application becomes more challenging. We further investigate how effective the learning to select framework is for selectively applying a query independent feature when there are more than two candidate features. In particular, we compare the learning to select with a simulated query type prediction approach, which knows with certainty the query type before applying a query independent feature (Section 6.2.3.3).

6.2.2 Experimental Settings

We use the TREC 2003 and 2004 Web track datasets, which contain three types of topic, namely homepage finding (HP), named page finding (NP) and topic distillation (TD) topics. This allows us to simulate an oracle query type prediction approach, which is used as a baseline in our third research question (RQ3). A breakdown of these topics per topic type is presented in Table 6.1.

In particular, the TREC 2003 and 2004 Web track are based on the TREC .GOV test collection. For indexing and retrieval, we use the Terrier IR platform[1] (Ounis et al., 2006), and apply standard stopwords removal. In addition, to boost early precision, we apply the first two steps of the Porter's stemming algorithm for English. We index the body, anchor text and titles of documents as separate fields

[1]`http://www.terrier.org`

and use the PL2F field-based DFR document weighting model (Equation (2.22)), which has shown its effectiveness in various retrieval tasks (Hannah et al., 2007; Lioma et al., 2006). Moreover, to build a strong baseline, we apply term dependency (described in Section 2.5.2) into the PL2F weighting scheme, denoted as PL2FProx. In particular, we employ the full dependency as it generally performs better than the sequential dependency (Peng et al., 2007). We experiment with the three query independent features introduced in Section 2.6, namely PageRank (PR), URL depth (UD) and click distance (CD), which have shown their effectiveness in various applications (Craswell et al., 2005; Kamps et al., 2004; Metzler et al., 2005; Peng & Ounis, 2009; Peng et al., 2007). For example, URL depth is very effective for finding a homepage (Kamps et al., 2004).

In search applications, users do not specify the type of their submitted query. In order to simulate a real IR environment, we mix the three topic types together. As in the training procedure in Section 5.3.4, we divide each of the two datasets (Table 6.1) into five equal size folds, testing the framework on one fold, using another as a validation set and the other three as a training set.

The evaluation measure used in all our experiments is the mean average precision (MAP). The parameters that are related with the PL2F document weighting model, the term dependency model, and the FLOE methods are set by optimising MAP on the training dataset, using a simulated annealing procedure (Skiścim & Golden, 1983). We use $FLOE^{+}$ (Equation (2.77)) for PageRank and $FLOE^{-}$ (Equation (2.78)) for the URL depth and click distance. The number of the top ranked documents (T) and the number of neighbouring queries (K), introduced in Chapter 4, are also set by optimising MAP over the training dataset, using a large range of different value settings. For the click distance feature, we use `firstgov.gov` as the root. The maximum Click Distance is 46 in the .GOV collection. For those documents that cannot be reached from the root, we assume a Click Distance of 47. In addition, six variants (namely KNN-Rel, KNN-KL, KNN-JS, Kmeans-Rel, Kmeans-KL, and Kmeans-JS) derived from the learning to select framework are investigated in the experiments. These variants have been described in detail in Section 4.4.3. We report the obtained results and their analysis in the following section.

6.2.3 Experimental Results

In this section, we investigate three different research questions that are defined in Section 6.2.1. Section 6.2.3.1 investigates how important it is to selectively apply an appropriate query independent feature. The effectiveness of the learning to select framework for selecting an appropriate query independent feature from two candidate document features and from more than two candidate document features are investigated in Section 6.2.3.2 and Section 6.2.3.3, respectively.

6.2.3.1 The Importance of Selective Application

To assess the benefit that could be received by selectively integrating an appropriate query independent document feature into a document weighting model, We produce the upper bounds of the retrieval performance by manually selecting the most effective query independent feature on a per-query basis. This allows us to estimate the extent to which it is indeed possible to enhance the retrieval performance of an IR system when the most appropriate query independent feature is applied on a per-query basis.

Table 6.2 provides the MAP upper bounds that can be achieved by manually and selectively applying a query independent feature on a per-query basis, first when there are two possible candidate features (rows 5-7), and second when we use all three features (row 8). PR|UD, PR|CD, UD|CD, and PR|UD|CD denote that the selective application is conducted between PageRank and URL depth, PageRank and click distance, URL depth and click distance, and Pagerank, URL depth and click distance, respectively. In each column, values that are statistically different from PL2FProx, PL2FProx+PR, PL2FProx+UD and PL2FProx+CD are marked with $*$, $\diamond$, $\star$ and $\bullet$, respectively (Wilcoxon Matched-Pairs Signed-Ranks Test, $p < 0.05$).

From Table 6.2, it is clear that using a manual selective method leads to significant increases in performance compared to the PL2FProx baseline, as well as to systems where a query independent feature was applied uniformly to all queries. We also observe that the upper bounds of the selective application among three query independent features are markedly higher than the selective application between any two of them, although not significantly so. This suggests that the

		TREC 2003	TREC 2004
1	PL2FProx	0.6066	0.4661
2	+PR	0.6541	0.5231
3	+UD	0.6359	0.5092
4	+CD	0.6216	0.5170
5	+(PR|UD)	**0.6889** $* \diamond \star$	**0.5743** $* \diamond \star$
6	+(PR|CD)	**0.6688** $* \diamond \bullet$	**0.5620** $* \diamond \bullet$
7	+(UD|CD)	**0.6616** $* \star \bullet$	**0.5699** $* \star \bullet$
8	+(PR|UD|CD)	**0.6934** $* \diamond \star \bullet$	**0.5914** $* \diamond \star \bullet$

Table 6.2: The MAP upper bounds, highlighted in bold, which are achieved by the manual selective application of a query independent feature on the TREC 2003 and TREC 2004 datasets.

selective application of a query independent feature on a per-query basis is very important for a Web search system, and that the retrieval performance could be further improved when the number of query independent features increases.

6.2.3.2 Learning to Select with Two Candidates

We test how effective our proposed learning to select framework is for selectively applying a query independent feature when there are two candidate features. In order to do so, we compare our proposed method to the PL2FProx baseline, as well as to the retrieval method that applies a query independent feature uniformly to all queries.

Tables 6.3, 6.4 and 6.5 show the MAP obtained by applying our proposed learning to select framework when there are two candidate features. The best retrieval performance in each MAP column is highlighted in bold. The symbol $*$ denotes that the MAP obtained by using learning to select is statistically better than the one achieved by the PL2FProx baseline, as well as all the systems where a query independent feature has been uniformly applied to all queries, according to the Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$). *Number* reports the number of queries for which the selected query independent feature has been correctly applied (denoted *Pos.*), using the manual upper bound approach as a ground truth. Conversely, the column *Neg.* reports the number of queries for which the system has failed to apply the most appropriate feature. The column *Neu.* reports the number of queries where all query independent features produced the same MAP. The symbol † denotes that the learning to select framework applies the most appropriate query independent feature for a statistically significant number of queries, according to the Sign Test ($p < 0.05$).

From Tables 6.3, 6.4 and 6.5, we can see that, for the three different combinations, namely PR|UD, PR|CD and UD|CD, the learning to select framework can always markedly improve on the PL2FProx baseline and on that of the systems where a query independent feature is uniformly applied. In particular, for the selective application between PageRank and URL depth, the improvement is constantly statistically significant on both datasets by using our proposed query features (i.e. KL and JS). For example, in Table 6.3 and on the TREC 2003 dataset, 0.6066 (PL2FProx) → 0.6809 (KNN-JS), 0.6541 (+PR) → 0.6809 (KNN-JS), and 0.6459 (+UD) → 0.6809 (KNN-JS). In addition, by comparing the two different neighbouring query finding techniques, we observe that both KNN and k-means are effective. For example, in Table 6.3 and on TREC 2003 dataset, 0.6541 (+PR) → 0.6809 (KNN-JS) and 0.6541 (+PR) → 0.6776 (Kmeans-JS). Moreover, the retrieval performances obtained by using KNN and k-means are close. For example, in Table 6.3 and on the TREC 2004 dataset, 0.5566 (KNN-KL) vs. 0.5554 (Kmeans-KL). Furthermore, we observe that a statistically significant number of queries have been applied with the most appropriate query independent feature on all possible combinations and on both TREC 2003 and TREC 2004 datasets.

This suggests that learning to select is an effective method for selecting an appropriate feature from any two candidate features. In particular, our proposed use of a divergence measure as a query feature is more effective than the use of the mean of the relevance scores. Besides, both KNN and k-means are effective in finding neighbouring queries.

6.2.3.3 Learning to Select with Three Candidates

As the number of candidate features increases, the selective application method raises more challenges. We further investigate how effective our proposed learning to select framework is for selectively applying the most appropriate query independent feature when there are more than two candidate features. In particular,

	TREC 2003				TREC 2004			
		Number				Number		
	MAP	Pos.	Neg.	Neu.	MAP	Pos.	Neg.	Neu.
PL2FProx	0.6066	-	-	-	0.4661	-	-	-
+PR	0.6541	-	-	-	0.5231	-	-	-
+UD	0.6359	-	-	-	0.5092	-	-	-
Learning to Select								
KNN-Rel	0.6679 †	100	50	200	0.5476 †	90	54	81
KNN-KL	0.6794 † ∗	111	39	200	**0.5566** † ∗	101	43	81
KNN-JS	**0.6809** † ∗	107	43	200	0.5559 † ∗	103	41	81
Kmeans-Rel	0.6616 †	98	52	200	0.5435 †	89	55	81
Kmeans-KL	0.6771 † ∗	107	43	200	0.5554 † ∗	102	42	81
Kmeans-JS	0.6776 † ∗	108	42	200	0.5539 † ∗	100	44	81

Table 6.3: Evaluation of the learning to selective framework for selectively applying a query independent feature between PageRank (PR) and URL depth (UD).

	TREC 2003				TREC 2004			
		Number				Number		
	MAP	Pos.	Neg.	Neu.	MAP	Pos.	Neg.	Neu.
PL2FProx	0.6066	-	-	-	0.4661	-	-	-
+PR	0.6541	-	-	-	0.5231	-	-	-
+CD	0.6216	-	-	-	0.5170	-	-	-
Learning to Select								
KNN-Rel	0.6570 †	90	47	213	0.5312 †	89	47	89
KNN-KL	0.6636 †	95	42	213	0.5363 †	95	41	89
KNN-JS	**0.6649** †	96	41	213	0.5363 †	95	41	89
Kmeans-Rel	0.6573 †	89	48	213	0.5315 †	88	48	89
Kmeans-KL	0.6647 †	97	40	213	**0.5378** †	96	40	89
Kmeans-JS	0.6648 †	97	40	213	**0.5378** †	96	40	89

Table 6.4: Evaluation of the learning to selective framework for selectively applying a query independent feature between PageRank (PR) and click distance (CD).

	TREC 2003				TREC 2004			
		Number				Number		
	MAP	Pos.	Neg.	Neu.	MAP	Pos.	Neg.	Neu.
PL2FProx	0.6066	-	-	-	0.4661	-	-	-
+UD	0.6359	-	-	-	0.5092	-	-	-
+CD	0.6216	-	-	-	0.5170	-	-	-
Learning to Select								
KNN-Rel	0.6432 †	90	56	204	0.5317 †	84	55	86
KNN-KL	0.6500 †	99	47	204	**0.5345** †	90	49	86
KNN-JS	0.6497 †	97	49	204	**0.5345** †	90	49	86
Kmeans-Rel	0.6480 †	91	55	204	0.5198 †	82	57	86
Kmeans-KL	0.6485 †	100	46	204	0.5338 †	89	50	86
Kmeans-JS	**0.6505** †	99	47	204	0.5336 †	89	50	86

Table 6.5: Evaluation of the learning to selective framework for selectively applying a query independent feature between URL depth (UD) and click distance (CD).

we select the most appropriate query independent feature out of the three used PR, UD, and CD features.

Table 6.6 shows the MAP obtained by applying our proposed learning to select framework when there are more than two candidate features. The best retrieval performance in each MAP column is highlighted in bold. The symbol † denotes that the learning to select framework applies the most appropriate query independent feature for a statistically significant number of queries, according to the Sign Test ($p < 0.05$). The symbol $*$ denotes that the MAP obtained by using learning to select is statistically better than the one achieved by the PL2FProx baseline, as well as all the systems where a query independent feature has been uniformly applied to all queries, according to the Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$).

The evaluation results from Table 6.6 show that our proposed learning to select framework can constantly make a significant improvement over PL2FProx and that of the systems where a query independent feature was uniformly applied, by using our proposed query features (i.e. KL and JS). The observation is upheld on both datasets and in consistency with what we found in Section 6.2.3.2, namely KL and JS are more effective than Rel.

Moreover, we also observe that a statistically significant number of queries have been applied with the most appropriate query independent feature on both the TREC 2003 and the TREC 2004 datasets. In addition, comparing the best MAP results (highlighted in bold) that can be obtained in each dataset in Tables 6.3, 6.4, 6.5, and 6.6, we can see that the retrieval performance obtained by using our proposed learning to select framework can be further improved when there are more than two candidate query independent features. For example, on the TREC 2003 dataset, 0.6809 → 0.6864, 0.6649 → 0.6864, and 0.6505 → 0.6864. This is encouraging, as this suggests that our proposed framework remains effective and robust even when the number of candidate features increases. Overall, while the results obtained in Tables 6.3, 6.4, 6.5, and 6.6 are naturally lower than the upper bounds performances in Table 6.2, they are nevertheless roughly reasonably comparable.

In addition, since there are three types of queries on the Web track TREC 2003 and TREC 2004 datasets, we simulate a query type prediction (QTP) method, which applies the most appropriate query independent feature for a given query type, and compare it with the best performing variant of the learning to select framework. The simulation process is the same as we conducted in Section 5.3.3.

From Table 6.7, we can see that our proposed method constantly outperforms the QTP method in both accuracy and MAP on both datasets. This particularly stresses the effectiveness and robustness of our approach compared to the QTP method, given that the query type prediction in a practical system is usually much lower than 100% (see Section 3.6). It also suggests that queries which have the same type do not necessarily equally benefit from the application of a given query independent feature since the MAP value obtained from the QTP method is not equal to the value of the upper bound on each dataset, even though the accuracy of the query type prediction is simulated equal to 100%.

6.2.4 Summary

In this section, we deployed our proposed learning to select framework for the selective application of a query independent feature on a per-query basis. We tested the effectiveness of the learning to select framework on the TREC .GOV

	TREC 2003				TREC 2004			
		Number				Number		
	MAP	Pos.	Neg.	Neu.	MAP	Pos.	Neg.	Neu.
PL2FProx	0.6066	-	-	-	0.4661	-	-	-
+PR	0.6541	-	-	-	0.5231	-	-	-
+UD	0.6359	-	-	-	0.5092	-	-	-
+CD	0.6216	-	-	-	0.5170	-	-	-
Learning to Select								
KNN-Rel	0.6718 †	100	61	189	0.5501 †	97	57	71
KNN-KL	0.6843 † $*$	109	52	189	0.5610 † $*$	100	54	71
KNN-JS	0.6813 † $*$	103	58	189	0.5604 † $*$	103	51	71
Kmeans-Rel	0.6716 †	101	60	189	0.5457 †	94	60	71
Kmeans-KL	0.6859 † $*$	108	53	189	**0.5658** † $*$	103	51	71
Kmeans-JS	**0.6864** † $*$	108	53	189	0.5647 † $*$	104	50	71

Table 6.6: Evaluation of the learning to selective framework for selectively applying a query independent feature among PageRank (PR), URL depth (UD), and click distance (CD).

	MAP	Pos.	Neg.	Neu.	Accuracy
TREC 2003					
Kmeans-JS	**0.6864**	108	53	189	**67.08 %**
QTP	0.6752	103	58	189	63.97%
TREC 2004					
Kmeans-KL	**0.5658**	103	51	71	**66.88%**
QTP	0.5569	99	55	71	64.28%

Table 6.7: Comparison between the best performing variant of learning to select and the oracle QTP method for the selective application of a query independent feature among PageRank, URL depth, and click distance.

Web test collection and the mixed topic sets from the TREC 2003 and TREC 2004 Web tracks.

We obtained very encouraging experimental results. First, we showed that the retrieval performance can be significantly improved by an optimal selective application of a query independent feature (Section 6.2.3.1). This indicates that the selective application of the query independent feature on a per-query basis can indeed significantly enhance the retrieval performance of a Web IR system.

Second, using our proposed learning to select framework, and any two query independent features, we observed that the most appropriate feature has been applied for a statistically significant number of queries (see Section 6.2.3.2). The improvement in MAP was statistically significant when the selective application occurred between PageRank and URL depth, by using our proposed query features (see Table 6.3 in Section 6.2.3.2).

Third, as the number of candidate features increases, the selective application raises more challenges. Therefore, we further investigated how effective our proposed framework is for selectively applying the most appropriate query independent feature when there are more than two candidate features. The experimental results showed that our proposed learning to select framework can make further improvement to the retrieval performance, when the number of candidate features increases. We also observed that the most appropriate query independent feature has been applied in a statistically significant number of queries. Moreover, we compared our proposed method to a simulated QTP method, which has an ideal 100% accuracy on the query type prediction. We observed that our proposed framework constantly outperforms the QTP method on both datasets. This suggests that our proposed learning to select framework is effective and robust.

By comparing the retrieval performances obtained by the six variants of the learning to select framework, we observed that both KNN and k-means are effective in finding neighbouring queries for Web search. In addition, the divergence measures are more effective than the mean of the relevance scores, which ignores the distribution of relevance scores. This observation is consistent with the findings of Chapter 5.

6.3 Web Search Diversification

Users' information need is represented in the form of a query, which is often short and this makes it ambiguous to IR systems (Spärck Jones et al., 2007). For example, the query "terrier" can be interpreted as involving a topic related to "dog" or "IR platform". In such a situation, an accurate interpretation of the users' information need is hard to be determined.

The simplest approach could be to assume the provided query is well defined to satisfy users' information need and directly return the ranked list of retrieved documents for the given query. An alternative approach could be to infer the most plausible meaning underlying the given query (e.g. the most popular), and return the ranked documents which are related to the inferred particular meaning. Both approaches may answer the users' information need by chance, but could also fail in some cases, leading to users dissatisfaction. A different approach that has been deployed in many Web search engines is to propose several different interpretations of the underlying meaning for a given query and ask users for explicit feedback. However, not every user is willing to conduct this additional step for a simple search.

A more sensible approach is to return a ranked list of documents that provide a complete coverage for a query, while avoiding excessive redundancy in the result list (Clarke et al., 2009). This is called a diversity search task and several different approaches have been proposed in the Web track 2009 (Balog et al., 2009; Chandar et al., 2009; Craswell et al., 2009). In particular, a novel framework called xQuAD (eXplicit Query Aspect Diversification) (McCreadie et al., 2009; Santos et al., 2010) has achieved the best retrieval performance on different evaluation measures on the large-scale TREC ClueWeb09 category B collection (Clarke et al., 2009). Moreover, one of the most important components of xQuAD called sub-query importance estimator has many variants, the details of which will be introduced in Section 6.3.1. In this section, we deploy and test our proposed learning to select framework for selectively applying an appropriate sub-query importance estimator variant in the xQuAD framework on a per-query basis.

The remainder of this section is organised as follows: Section 6.3.1 introduces the xQuAD framework; we present our research questions in Section 6.3.2; Section 6.3.3 describes the settings of our experiments; The experimental results obtained and their analysis are discussed in Section 6.3.4. We draw conclusions on the application of our proposed learning to select framework for the Web search diversity task in Section 6.3.5.

6.3.1 The xQuAD Framework

The xQuAD framework (Santos et al., 2010) is centred around the concept of sub-queries, which is inspired by the greedy approximation approach to the general diversification problem (Carbonell & Goldstein, 1998). However, in contrast to other approaches in the literature, xQuAD performs an explicit diversification of the documents retrieved for a given query, by exploiting the relation between these documents and the aspects uncovered from this query in the form of sub-queries. For example, the first query of Web track 2009 is "obama family tree", which has three different sub-queries: "Find the TIME magazine photo essay - Barack Obama's Family Tree", "Where did Barack Obama's parents and grandparents come from", and "Find biographical information on Barack Obama's mother".

In the xQuAD framework, there is an important component called *sub-query importance estimator*, which models the relative importance of different query aspects that are represented as different sub-queries. In particular, three different ways of computing the importance of a sub-query have been proposed along with the xQuAD framework (Santos et al., 2010).

The first estimator (denoted U) considers each sub query with equal weight, given as follows:

$$i_U(s, q)_{s \in Q(q)} = \frac{1}{|Q(q)|} \tag{6.1}$$

where $Q(q)$ is the sub query set that is associated with the initial query q.

Based on the idea that the relative importance of each sub-query can be discovered from how well it is covered by a given collection, the second estimator (denoted W) is defined as follows:

$$i_W(s, q)_{s \in Q(q)} = \frac{n(s)}{\sum_{s_i \in Q(q)} n(s_i)} \tag{6.2}$$

where $n(s)$ is the estimated number of documents that are retrieved for the sub query s by using a commercial search engine.

The third estimator (denoted C) is inspired by resource selection techniques in distributed information retrieval (Callan, 2000): Central Ranked-based Collection Selection (CRCS) (Shokouhi, 2007). CRCS not only ranks resources according to their estimated sizes but also considers the ranking position of each sampled document in the centralised ranking of resource descriptions. The basic idea is that a higher ranked document should convey more importance to its resource than a document appearing lower. Inspired by CRCS, the sub query importance estimator is given as follows:

$$i_C(s,q)_{s \in Q(q)} = \frac{n(s)}{max_{s_i \in Q(q)} n(s_i)} \times \frac{1}{\hat{n}(s)} \sum_{d|r(d,s)>0} \tau - j(d,q) \tag{6.3}$$

where $n(s)$ is the total number of documents retrieved for s, $\hat{n}(s)$ is the number of documents associated to the sub-query s that are among the top τ ranked documents for the initial query q, $j(d,q)$ is the ranking position of the document d with respect to q. More details about the xQuAD framework can be found in (Santos et al., 2010).

6.3.2 Research Questions

By deploying our proposed learning to select framework for the Web diversity search task, we tackle the following three research questions:

- RQ1, we assess how important it is to selectively apply the sub-query importance estimator in the xQuAD framework for the Web search diversity task (Section 6.3.4.1).
- RQ2, we investigate the effectiveness of our proposed learning to select framework for the selective application of the sub-query importance estimator. In particular, the investigation is conducted across three different document weighting schemes (Section 6.3.4.2).
- RQ3, we test the importance of the query feature and neighbouring query finding components in our proposed learning to select framework (Section 6.3.4.3).

6.3.3 Experimental Settings

We conduct our experiments on the newly released TREC ClueWeb09 dataset[1], which was constructed by crawling the Web during January and February 2009, and contains roughly 1 billion web pages. In our experiments, we use the subset of this collection (also called "Category B"), which is used in the TREC 2009 Web diversity task and consists of about 50 million English-language documents. For evaluation, we use the latest TREC 2009 Web track dataset, which has 50 query topics with relevance assessments that were created and assessed by National Institute of Standards and Technology (NIST). Each topic has 3 to 8 sub-topics and with a mean of 4.9.

Similar to the settings in Section 6.2.2, we use the Terrier IR platform (Ounis et al., 2006) for indexing and retrieval. All corpora are indexed by removing standard stopwords and applying Porter's stemming algorithm for English. We use three different document weighting models, namely BM25 (Equation (2.6)), Hiemstra's language modelling (LM) (Equation (2.12)) and the parameter-free DPH Divergence From Randomeness model (Equation (2.21)), which are the representatives of different probabilistic retrieval families.

The evaluation measures for the new Web diversity task are $\alpha-$normalised discounted cumulative gain ($\alpha-$NDCG (Clarke et al., 2008)) and intent-aware precision (IA-P (Agrawal et al., 2009)), which are described in Section 2.7. As before, we divide the dataset into 5 folds of equal size and iteratively test our learning to select framework on one fold by using another fold as a validation dataset and the remaining three folds as a training dataset. The number of the top ranked documents (T) and the K value in Chapter 4, are set by optimising $\alpha-$NDCG@10 during the training process, using a large range of different value settings. The parameters that are related with the BM25 and LM document weighting models are also set by optimising $\alpha-$NDCG@10, using a simulated annealing procedure (Skiścim & Golden, 1983). DPH is a parameter-free model, therefore it does not require training.

To deploy our proposed learning to select framework for selectively applying an appropriate sub-query importance estimator for the Web diversity task, we use the

[1]`http://boston.lti.cs.cmu.edu/Data/clueweb09/`

ranking function that is obtained by using a document weighting model only (e.g. BM25) as the base ranking function (see Section 4.4.1), and the ranking function that is obtained by using one of the sub-query importance estimator variant in the xQuAD framework as the candidate ranking function (see Section 4.4.1).

6.3.4 Experimental Results

In this section, we tackle three different research questions, which are defined in Section 6.3.2. Section 6.3.4.1 investigates the importance of selectively applying a sub-query importance estimator on a per-query basis. In Section 6.3.4.2, we investigate the effectiveness of the learning to select framework in selecting an appropriate sub-query importance estimator for each given query in three different document weighting schemes. The importance of each component of the learning to select framework is investigated in Section 6.3.4.3.

6.3.4.1 Importance of Selective Application

In order to assess how important it is to selectively apply an appropriate sub-query importance estimator in the xQuAD framework on a per-query basis for the Web diversity task, we simulate an oracle system that applies the best performing estimator for each query. This allows us to know the extent to which it is indeed possible to enhance the diversification effectiveness when the sub-query importance estimator is correctly applied on a per-query basis.

Table 6.8 provides the evaluation of the systematic application of a sub-query importance estimator, and the simulated selective application of an appropriate sub-query importance estimator with 100% accuracy, using different weighting schemes and using different evaluation measures. xQuAD$_U$, xQuAD$_W$ and xQuAD$_C$ denote that we systematically apply the U estimator (Equation (6.1)), the W estimator (Equation (6.2)), and the C estimator (Equation (6.3)) in the xQuAD framework for diversity search, respectively. From the results in Table 6.8, we observe that the systematic application of the sub-query importance estimator can constantly make marked improvements on the diversification performance across DPH, BM25 and LM, using $\alpha-$NDCG at different levels, e.g., in $\alpha-$NDCG@10 on the DPH weighting scheme: 0.2121 $\rightarrow$ 0.2551; 0.2121 $\rightarrow$

	$\alpha-$NDCG			IA-P		
	@5	@10	@100	@5	@10	@100
DPH	0.1983	0.2121	0.3042	0.1090	0.1064	0.0621
+xQuAD$_U$	0.2217	0.2551	0.3390	0.1113	0.1073	0.0611
+xQuAD$_W$	0.2178	0.2373	0.3265	0.1158	0.1043	0.0606
+xQuAD$_C$	0.2381	0.2522	0.3392	0.1105	0.1002	0.0548
Upper Bound	**0.2780** $*\diamond\star$	**0.2968** $*\diamond\star$	**0.3747** $*\diamond\star$	**0.1412** $*\diamond\star$	**0.1290** $*\diamond\star$	**0.0663** $*\diamond\star$
BM25	0.1868	0.2126	0.3063	0.0881	0.0838	0.0634
+xQuAD$_U$	0.2543	0.2785	0.3614	0.1256	0.1120	0.0614
+xQuAD$_W$	0.2528	0.2630	0.3583	0.1218	0.0994	0.0611
+xQuAD$_C$	0.2430	0.2643	0.3462	0.1167	0.1035	0.0515
Upper Bound	**0.3033** $*\diamond\star$	**0.3161** $*\diamond\star$	**0.3933** $*\diamond\star$	**0.1445** $*\diamond\star$	**0.1252** $*\diamond\star$	**0.0636** $*\diamond\star$
LM	0.0925	0.1064	0.1895	0.0495	0.0478	0.0339
+xQuAD$_U$	0.1142	0.1288	0.2148	0.0567	0.0497	0.0358
+xQuAD$_W$	0.1064	0.1146	0.2011	0.0540	0.0478	0.0343
+xQuAD$_C$	0.1539	0.1721	0.2562	0.0750	0.0597	0.0445
Upper Bound	**0.1752** $*\diamond\star$	**0.1861** $*\diamond\star$	**0.2773** $*\diamond\star$	**0.0875** $*\diamond\star$	**0.0744** $*\diamond\star$	**0.0523** $*\diamond\star$

Table 6.8: The highest score based on each weighting scheme is highlighted in bold and scores that are statistically better than the corresponding xQuAD$_U$, xQuAD$_W$, and xQuAD$_C$ are marked with $*$, $\diamond$, and $\star$, respectively (Wilcoxon matched-pairs signed-ranks test, $p < 0.05$).

0.2373; and 0.2121 → 0.2522. Whereas, for the IA-P measure, in some cases, the obtained retrieval performances are decreased after applying the xQuAD framework, e.g., in IA-P@100 on the DPH weighting scheme: 0.0621 → 0.0611; 0.0621 → 0.0606; and 0.0621 → 0.0548.

However, by selectively applying the xQuAD framework with an appropriate sub-query importance estimator, the obtained retrieval performances constantly outperform the DPH, BM25, and LM ranking baselines, using all evaluation measures and at all levels, e.g., using $\alpha-$NDCG on the DPH weighting scheme: 0.1983 → 0.2780 when @5; 0.2121 → 0.2968 when @10; and 0.3042 → 0.3747 when @100. Furthermore, compared to the systematic application of a sub-query importance estimator, the obtained retrieval performance is always significantly boosted, e.g., using $\alpha-$NDCG@10 on the DPH weighting scheme: 0.2551 → 0.2968; 0.2373 → 0.2968; and 0.2522 → 0.2968. The above observations show that, indeed, different sub-query importance estimators do favour different queries and that the appropriate selective application of a sub-query importance estimator could enhance the diversification performance.

	$\alpha-$NDCG			IA-P		
	@5	@10	@100	@5	@10	@100
DPH+xQuAD$_U$	0.2217 α	<u>0.2551</u>	0.3390	0.1113	<u>0.1073</u>	<u>0.0611</u>
DPH+xQuAD$_W$	0.2178 β	0.2373 β	0.3265	<u>0.1158</u>	0.1043	0.0606
DPH+xQuAD$_C$	<u>0.2381</u>	0.2522	<u>0.3392</u>	0.1105	0.1002 γ	0.0548
Learning to select						
KNN-Rel	0.2373	0.2644 $\uparrow$	0.3468 $\uparrow$	0.1186 $\uparrow$	0.1102 $\uparrow$	0.0564
KNN-KL	0.2521 $\uparrow$	**0.2760** $\uparrow$	0.3539 $\uparrow$	0.1198 $\uparrow$	**0.1128** $\uparrow$	0.0574
KNN-JS	0.2510 $\uparrow$	0.2750 $\uparrow$	0.3537 $\uparrow$	0.1188 $\uparrow$	0.1123 $\uparrow$	0.0574
Kmeans-Rel	0.2360	0.2596 $\uparrow$	0.3415 $\uparrow$	0.1159 $\uparrow$	0.1082 $\uparrow$	0.0544
Kmeans-KL	**0.2544** $\uparrow$	0.2679 $\uparrow$	**0.3558** $* \uparrow$	**0.1225** $\uparrow$	0.1119 $\uparrow$	0.0577
Kmeans-JS	0.2519 $\uparrow$	0.2710 $\uparrow$	0.3550 $\uparrow$	0.1210 $\uparrow$	0.1096 $\uparrow$	**0.0582**

Table 6.9: Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the DPH weighting scheme.

6.3.4.2 Effectiveness of the LTS Framework

In order to test the effectiveness of our proposed learning to select framework for selectively applying an appropriate sub-query importance estimator for a given query, we compare it with the diversification performances that are obtained by using xQuAD with different sub-query importance estimators, which are systematically and equally applied to all queries.

Tables 6.9, 6.10, and 6.11 present the evaluation of the diversification performances obtained by using xQuAD with different sub-query importance estimators and by applying our proposed LTS framework in terms of $\alpha-$NDCG and IA-P at different levels, using the DPH, BM25 and LM weighting schemes, respectively.

The best diversification performance obtained by systematically applying a sub-query importance estimator on all queries is highlighted with underline. $\uparrow$ denotes that the obtained diversification performance by using our proposed LTS framework is better than the underlined score. The $*$ symbol indicates that the retrieval performance obtained by using our proposed LTS framework is significantly better than the underlined score, according to the Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$). The best diversification performances for each evaluation measure are emphasised in bold. The α, β, and γ symbols indicate that the retrieval performance obtained by the best of our proposed LTS framework is significantly better than the xQuAD$_U$, xQuAD$_W$, and xQuAD$_C$, respectively.

	$\alpha-$NDCG			IA-P		
	@5	@10	@100	@5	@10	@100
BM25+xQuAD$_U$	<u>0.2543</u> α	<u>0.2785</u>	<u>0.3614</u>	<u>0.1256</u>	<u>0.1120</u>	<u>0.0614</u>
BM25+xQuAD$_W$	0.2528 β	0.2630 β	0.3583	0.1218	0.0994 β	0.0611
BM25+xQuAD$_C$	0.2430 γ	0.2643	0.3462 γ	0.1167 γ	0.1035	0.0515 γ
Learning to select						
KNN-Rel	0.2616 $\uparrow$	0.2823 $\uparrow$	0.3614	0.1284 $\uparrow$	0.1099	0.0592
KNN-KL	0.2711 $\uparrow$	0.2926 $\uparrow$	0.3749 $\uparrow$	0.1309 $\uparrow$	**0.1140** $\uparrow$	0.0605
KNN-JS	0.2686 $\uparrow$	0.2897 $\uparrow$	0.3707 $\uparrow$	0.1250	0.1085	0.0594
Kmeans-Rel	0.2657 $\uparrow$	0.2842 $\uparrow$	0.3695 $\uparrow$	0.1291 $\uparrow$	0.1101	0.0597
Kmeans-KL	**0.2834** $* \uparrow$	**0.3021** $\uparrow$	0.3774 $\uparrow$	**0.1323** $\uparrow$	0.1121 $\uparrow$	**0.0616** $\uparrow$
Kmeans-JS	0.2825 $\uparrow$	0.3010 $\uparrow$	**0.3782** $\uparrow$	0.1267 $\uparrow$	0.1053	0.0570

Table 6.10: Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the BM25 weighting scheme.

In addition, as defined in Section 4.4.3, KNN-Rel, KNN-KL and KNN-JS denote that we employ the KNN algorithm for identifying similar queries based on the query feature of the mean of the relevance scores, the KL divergence score and the JS divergence score respectively; Kmeans-Rel, Kmeans-KL and Kmeans-JS denote that we use the Kmeans algorithm for identifying similar queries based on the query feature of the mean of the relevance scores, the KL divergence score, and the JS divergence score, respectively.

From the results in Tables 6.9, 6.10, and 6.11, we observe that the best diversification performance in each column is achieved by using our proposed LTS framework, e.g., using the DPH weighting scheme (Table 6.9), we obtain 0.2544 in $\alpha-$NDCG@5, 0.2760 in $\alpha-$NDCG@10 and 0.3558 in $\alpha-$NDCG@100. Moreover, the best performing results always make marked improvements over the systematical application approaches. In particular, such improvements are statistically significant in some cases, e.g., using the language modelling approach (Table 6.11) and using $\alpha-$NDCG@10: 0.1288 $\rightarrow$ 0.1822 and 0.1146 $\rightarrow$ 0.1822.

Furthermore, compared to the best performing sub-query importance estimator (highlighted with underline), the diversification performances obtained by using our proposed LTS framework, with various neighbouring querying finding algorithms and various query features, make marked improvement in most cases, e.g., using DPH weighting scheme (Table 6.9) and using $\alpha-$NDCG@10: 0.2551 $\rightarrow$

	$\alpha-$NDCG			IA-P		
	@5	@10	@100	@5	@10	@100
LM+xQuAD$_U$	0.1142 α	0.1288 α	0.2148 α	0.0567 α	0.0497 α	0.0358 α
LM+xQuAD$_W$	0.1064 β	0.1146 β	0.2011 β	0.0540 β	0.0478 β	0.0343 β
LM+xQuAD$_C$	<u>0.1539</u>	<u>0.1721</u>	<u>0.2562</u>	<u>0.0750</u>	<u>0.0597</u>	<u>0.0445</u>
Learning to select						
KNN-Rel	0.1512	0.1676	0.2544	0.0724	0.0588	0.0460 $\Uparrow$
KNN-KL	0.1668 $\Uparrow$	**0.1822** $\Uparrow$	0.2632 $\Uparrow$	**0.0815** $\Uparrow$	**0.0664** $\Uparrow$	0.0466 $\Uparrow$
KNN-JS	**0.1682** $*$ $\Uparrow$	0.1819 $\Uparrow$	**0.2655** $\Uparrow$	0.0805 $\Uparrow$	0.0650 $\Uparrow$	0.0464 $\Uparrow$
Kmeans-Rel	0.1643 $\Uparrow$	0.1776 $\Uparrow$	0.2621 $\Uparrow$	0.0781 $\Uparrow$	0.0633 $\Uparrow$	0.0462 $\Uparrow$
Kmeans-KL	0.1657 $\Uparrow$	0.1816 $\Uparrow$	0.2640 $\Uparrow$	0.0791 $\Uparrow$	0.0645 $\Uparrow$	0.0462 $\Uparrow$
Kmeans-JS	0.1640 $\Uparrow$	0.1821 $\Uparrow$	0.2640 $\Uparrow$	0.0801 $\Uparrow$	0.0659 $\Uparrow$	**0.0467** $\Uparrow$

Table 6.11: Evaluation of the selective application of the sub-query importance estimator on the TREC Web diversity task using the language modelling weighting scheme.

0.2644; 0.2551 → 0.2760; 0.2551 → 0.2750; 0.2551 → 0.2596; 0.2551 → 0.2679; and 0.2551 → 0.2710. In particular, such an improvement is consistent using $\alpha-$NDCG@10 with the only exception being the use of the language modelling weighting scheme with the mean of the relevance scores as a query feature.

The above observations suggest that our proposed LTS framework is effective in applying an appropriate sub-query importance estimator on a per-query basis for the Web search diversity task.

6.3.4.3 Importance of the Query Feature and the Neighbouring Query Finding Algorithm

It is of note that there are two key components in the LTS framework: the query feature and the neighbouring object finding algorithm. In particular, we have investigated three different query features: namely the mean of the relevance scores, the KL divergence, and the JS divergence; and two different neighbouring query finding algorithms: KNN and *k*-means.

By using different query features and different neighbouring object finding algorithms for identifying similar queries in the LTS framework, from Tables 6.9, 6.10, and 6.11, we observe that the JS divergence measure and the KL divergence measure are producing very close retrieval performances, which is explained in that they are mathematically related. This observation is in line with the summary in Section 5.7. In addition, both divergence measures outperform the mean of the relevance scores in most cases. For example, using the BM25 weighting scheme (Table 6.10) and using $\alpha-$NDCG@10: 0.2823 $\rightarrow$ 0.2926; 0.2823 $\rightarrow$ 0.2897; 0.2842 $\rightarrow$ 0.3021; and 0.2842 $\rightarrow$ 0.3010. Furthermore, we also observe that both neighbouring object finding algorithms produce very close results, e.g., in the DPH weighting scheme and in $\alpha-$NDCG@10: 0.2760 versus 0.2679; and 0.2750 versus 0.2710. This is in consistency with the observation found in Chapter 5.

The above observations suggest that our proposed use of divergence measures as a query feature and the use of KNN and Kmeans as neighbouring query finding algorithms for identifying neighbouring queries are very effective.

6.3.5 Summary

In this section, we have deployed the learning to select framework for selectively applying a sub-query importance estimator in the xQuAD framework for the Web search diversity task. In addition, a full study of the effectiveness of our proposed learning to select framework in this application has been investigated on the TREC ClueWeb09 (category B) test collection and its corresponding TREC 2009 Web track topic set, using different weighting schemes.

Our experimental results showed that the diversification performance of the xQuAD framework can be significantly enhanced with the optimal selective application of a sub-query importance estimator (see Section 6.3.4.1). By comparing our proposed learning to select framework with several different baselines, we found that our proposed framework is effective and robust. Indeed, the learning to select framework always enhances the diversification performance across all weighting schemes (see Section 6.3.4.2).

In addition, by investigating the importance of query features and the neighbouring query finding algorithms, we found that our proposed use of divergence measures as a query feature is producing very close retrieval performance in two different neighbouring query finding algorithms. Moreover, both divergence measures are more effective than the mean of the relevance scores, which ignores the distribution of the document relevance scores (see Section 6.3.4.3). This is in line with the observations found in Chapter 5 and Section 6.2.

6.4 Enterprise Document Search

The IDC report from Feldman & Sherman (2003) quantifies the importance of information access within enterprises. The report suggests that not finding relevant information could result in poor decisions and lost sales because customers cannot find the required information on products or services and hence give up in frustration. The most common form of search within enterprises is document search. Compared to Web document search, there are several differences (Fagin et al., 2003):

- Enterprise documents are often created for the simple dissemination of information, rather than to attract and hold the attention of any specific group of users.
- A large fraction of queries tend to have a small set of correct answers (in most cases, there is only one relevant document that satisfies the user's information need), and the correct answer pages do not usually have any special characteristics.
- The number of documents in the Web is significantly larger than the number of documents in an enterprise. Documents in the Web are created by a large number of people while enterprise documents are created by a small number of individuals, for example, employees and IT contractors.

In general, the query expansion technique (see Section 2.4.1) helps to enhance the retrieval performance for enterprise document search (Balog et al., 2007). However, for some enterprise search queries, query expansion may fail. This is explained by the fact that an intranet collection generally reflects the view of the organisation that it serves and content generation often tends to be autocratic or bureaucratic rather than democratic (Fagin et al., 2003). This leads to a restricted use of alternative lexical representations, limiting the usefulness of query expansion. For these queries, it may be advantageous to use the well-known collection enrichment method (see Section 2.4.2), also called as external query expansion (Diaz & Metzler, 2006; Kwok & Chan, 1998), which performs query expansion using a larger and higher-quality external resource and then retrieves from the local collection using the expanded query.

Peng et al. (2009) proposed the use of query performance predictors to selectively apply collection enrichment (see Section 3.4). In particular, this approach uses a query performance predictor (see Section 2.4.3) to predict a given query's performance on both the internal and external resources. Then, it decides whether or not to apply CE based on the predicted performances. In this section, we deploy the learning to select framework for selectively applying an appropriate resource for expanding a given query, by setting the retrieval strategy without QE and CE as the base ranking function (see Section 4.4.1), and the retrieval strategy with the application of QE or CE as the candidate ranking function (see Section 4.4.1). For the remaining of this section, we introduce our research questions in Section 6.4.1. The settings for our experiments are presented in Section 6.4.2. Section 6.4.3 shows the experimental results and analysis. We draw conclusions about selectively applying an appropriate resource for expanding a query in Section 6.4.4.

6.4.1 Research Questions

In the following experiments, we address three main research questions:

- RQ1, we assess how important it is to selectively apply collection enrichment for enterprise search (Section 6.4.3.1).
- RQ2, we investigate the effectiveness of our proposed learning to select framework for selective CE. In particular, we use the query performance predictor-based approach, which was introduced in Section 3.4, as one of our baselines (Section 6.4.3.2).
- RQ3, we study the importance of the selected resource, which is used to expand an initial query. (Section 6.4.3.3).

6.4.2 Experimental Settings

We use the standard TREC Enterprise CERC test collection (Bailey et al., 2007), which is a crawl of the website of an Australian government research organisation, namely the Commonwealth Scientific and Industrial Research Organisation (CSIRO). This collection is a realistic setting for experimentation in enterprise search, with a real enterprise corpus and real user information needs. The collection contains research publications and reports, as well as Web sites devoted to the research areas of CSIRO. Approximately 7.9 million hyperlinks included in the collection, and 95% of pages have one or more outgoing links containing anchor text. For evaluation, we use the TREC 2007 and 2008 enterprise document search datasets, which are the last datasets used for the enterprise document search task and have 42 and 63 query topics with relevance assessments, respectively. These topics have title and narrative fields, however, for a realistic setting, we use query terms from only the title field.

We experiment with three different external resources, namely Wikipedia[1], Aquaint2[2], and .GOV (Craswell & Hawking, 2002). The Wikipedia corpus used in this paper is a snapshot from August 2008 and contains over 3 million articles written collaboratively by users worldwide. Articles in this collection cover a wide range of topics, including sports, historical events and science. The Aquaint2 collection consists of newswire information in English from six different sources, such as New York Times and Xinhua News Agency. The .GOV collection, which is also used in Section 6.2, is a crawl of the federal and state US government websites from early 2002, which includes $7,794$ hostnames, $11,164,829$ hyperlinks and $2,470,109$ cross-host hyperlinks. An overview of these collections can be found at Table 6.12. We use two different types of query performance predictors introduced in Section 2.4.3, including 6 pre-retrieval predictors (AvICTF, AvIDF, $\gamma1$, $\gamma2$, AvPMI and QS) and 3 post-retrieval predictors (CS, WIG and QF).

Similar to the settings in Sections 6.2.2 and 6.3.3, we use the Terrier IR platform (Ounis et al., 2006) for indexing and retrieval. All corpora are indexed by removing standard stopwords and applying Porter's stemming algorithm for

[1]`http://en.wikipedia.org/wiki/Wikipedia:Database_download`

[2]`http://trec.nist.gov/act_part/tracks/qa/qa.07.guidelines.html#documents`

Collection	**Description**	**# Docs**	**Fields**
CERC	TREC Enterprise Track	370,715	body, anchor text, title
Wikipedia	User-Generated Online Encyclopedia (August 2008)	3,588,998	body, title
Aquaint2	Newswire Articles	906,777	body, title
.GOV	TREC Corpus of US Government Websites	1,247,753	body, anchor text, title

Table 6.12: Overview of the test collections for expanding a given query.

English. We index the body, anchor text and titles of documents as separate fields. For Wikipedia, we ignore the anchor text field as our initial experiments found that it does not help to improve the retrieval performance, while the Aquaint2 newswire corpus does not have any anchor text. An overview of the fields applied for each collection is also included in Table 6.12. Documents are ranked using two different field-based document weighting model: PL2F (Equation (2.22)) and BM25F (Equation (2.7)).

Also, like the settings used in Sections 6.2.2 and 6.3.3, in the experiment, for each year's dataset, we divide it into 5 equal-size folds. We iteratively use one fold as test set, another fold as validate set and the remaining three folds as training set. The number of selected terms and the number of documents included in the pseudo-relevant set, namely $\#(term)$ and K_{doc} in Section 2.4, are set to 10 and 3, respectively, as suggested by (Amati, 2003). The number of the top ranked documents (T) and the K value in Chapter 4, are set by optimising MAP on the training dataset, using a large range of different value settings. The parameters that are related to the PL2F and BM25F document weighting models, the predictors and the threshold of the decision mechanism are also set by optimising MAP on the training dataset, using a simulated annealing procedure (Skiścim & Golden, 1983). Finally, for the post-retrieval predictors, the number of top ranked documents is also set by optimising MAP over the training dataset, using a large range of different value settings. The evaluation measures used in all our experiments are the Mean Average Precision (MAP) and the normalised Discounted Cumulative Gain (nDCG), which are the official evaluation measures of the enterprise document search task.

6.4.3 Experimental Results

In this section, we address three different research questions that are defined in Section 6.4.1. In Section 6.4.3.1, we assess the importance of choosing an appropriate resource for expanding a given query on a per-query basis. Section 6.4.3.2 investigates the effectiveness of the learning to select framework in selectively applying an appropriate resource for expanding a given query. In particular, the query performance predictor-based approach (Section 3.4) is used as one of our baselines. In Section 6.4.3.3, we study the importance of the selected resource for expanding a given query.

6.4.3.1 Importance of Selective CE

In order to assess how important it is to selectively apply collection enrichment on a per-query basis for enterprise document search, we simulate an oracle system that applies collection enrichment only for the queries where it is more beneficial to retrieval performance by applying collection enrichment compared to query expansion. This allows us to know the extent to which it is indeed possible to enhance the retrieval performance when collection enrichment is correctly applied on a per-query basis.

Tables 6.13 and 6.14 provide the evaluation of the systematic application of QE or CE and the retrieval performance upper bounds in the PL2F and BM25F document weighting schemes, respectively, in terms of MAP and nDCG on the TREC 2007 and TREC 2008 datasets. In each column, the highest value is highlighted in bold and the values that are statistically better than the systematic application of QE and CE are marked with $*$ (Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$)).

From the results in Tables 6.13 and 6.14, we observe that the systematic application of query expansion can constantly make marked improvements on the retrieval performance over PL2F and BM25F, using both the MAP and nDCG evaluation measures, e.g. the retrieval performance in MAP is increased from 0.4588 to 0.5015 on the TREC 2007 dataset using the PL2F weighting scheme. Moreover, the retrieval performance of the optimal selective application of collection enrichment leads to a further significant improvement over the systematic

TREC 2007 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	nDCG	MAP	nDCG	MAP	nDCG
PL2F	0.4588	0.7161	0.4588	0.7161	0.4588	0.7161
+QE	0.5015	0.7596	0.5015	0.7596	0.5015	0.7596
+CE	0.4859	0.7349	0.4747	0.7298	0.4777	0.7318
Upper Bound	**0.5241** $*$	**0.7762** $*$	**0.5164** $*$	**0.7697** $*$	**0.5185** $*$	**0.7710** $*$

TREC 2008 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	nDCG	MAP	nDCG	MAP	nDCG
PL2F	0.3564	0.5370	0.3564	0.5370	0.3564	0.5370
+QE	0.3695	0.5563	0.3695	0.5563	0.3695	0.5563
+CE	0.3629	0.5506	0.3504	0.5378	0.3544	0.5478
Upper Bound	**0.4058** $*$	**0.5913** $*$	**0.3976** $*$	**0.5745** $*$	**0.3954** $*$	**0.5782** $*$

Table 6.13: The MAP and nDCG results for applying QE and CE systematically using the PL2F document weighting model, and Upper Bound highlighted in bold, which is achieved by the optimal selective application of CE.

application of query expansion across three different external resources (e.g. on the TREC 2007 dataset and the Wikipedia external resource, the retrieval performance is significantly boosted from 0.5015 to 0.5241). This shows that, for some enterprise document search queries, it is indeed advantageous to use CE and the selective application of CE on a per-query basis is important for the enterprise document search.

6.4.3.2 Effectiveness of Selective CE

In this part, we investigate how effective our proposed learning to select framework is for selectively applying CE on a per-query basis, by comparing it with two different baselines: the first baseline is the document weighting model with the systematic application of QE. We use the application of QE as our baseline instead of CE given the observation from Tables 6.13 and 6.14, which shows that QE is consistently better than CE across three different external resources; the second baseline is the query performance predictor-based approach, which was described in Section 3.4.

Tables 6.15, 6.16, 6.17 and 6.18 present the evaluation of the selective appli-

TREC 2007 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	nDCG	MAP	nDCG	MAP	nDCG
BM25F	0.4587	0.7166	0.4587	0.7166	0.4587	0.7166
+QE	0.4847	0.7495	0.4847	0.7495	0.4847	0.7495
+CE	0.4755	0.7308	0.4646	0.7232	0.4656	0.7226
Upper Bound	**0.5095** ∗	**0.7703** ∗	**0.5058** ∗	**0.7686** ∗	**0.5093** ∗	**0.7669** ∗
TREC 2008 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	nDCG	MAP	nDCG	MAP	nDCG
BM25F	0.3366	0.5220	0.3366	0.5220	0.3366	0.5220
+QE	0.3423	0.5233	0.3423	0.5233	0.3423	0.5233
+CE	0.3382	0.5229	0.3225	0.5205	0.3325	0.5178
Upper Bound	**0.3760** ∗	**0.5524** ∗	**0.3633** ∗	**0.5482** ∗	**0.3670** ∗	**0.5435** ∗

Table 6.14: The MAP and nDCG results for applying QE and CE systematically using the BM25F document weighting model, and Upper Bound highlighted in bold, which is achieved by the optimal selective application of CE.

cation of collection enrichment on different external resources by using the query performance predictor-based and our proposed learning to select framework on the TREC 2007 and TREC 2008 datasets and in the PL2F and BM25F weighting schemes. The best retrieval performance in each column is highlighted in bold. The α, β, and γ symbols indicate that the retrieval performance obtained by the best of our proposed LTS framework is significantly better than PL2F (or BM25F), the systematic application of QE, and the systematic application of CE, respectively (Wilcoxon Matched-Pairs Signed-Ranks Test ($p < 0.05$)). The best retrieval performance obtained by the query performance predictor-based approach is emphasised with underline. $\uparrow$ denotes that the obtained retrieval performance by our proposed learning to select framework outperforms the underlined score. In particular, the $*$ symbol indicates that such improvement is statistically significant.

For the learning to select approach, six variants (defined in Section 4.4.3) are investigated in this section: KNN-Rel, KNN-KL and KNN-JS denote that we employ the KNN algorithm for identifying similar queries based on the query feature of the mean of the relevance scores, the KL divergence score and the JS divergence score respectively; Kmeans-Rel, Kmeans-KL and Kmeans-JS denote

TREC 2007 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
PL2F	0.4588 α	0.7161 α	0.4588 α	0.7161 α	0.4588 α	0.7161 α
+QE	0.5015 β	0.7596 β	0.5015 β	0.7596	0.5015 β	0.7596
+CE	0.4859 γ	0.7349 γ	0.4747 γ	0.7298 γ	0.4777 γ	0.7318 γ
Pre-retrieval Predictors						
AvICTF	0.4939	0.7434	0.4876	0.7429	0.4970	0.7569
AvIDF	0.4919	0.7422	0.4901	0.7451	0.4983	0.7589
Gamma1	0.5091	0.7621	0.5001	0.7564	0.5009	0.7573
Gamma2	0.5113	0.7645	0.5008	0.7562	0.4994	0.7561
AvPMI	0.5022	0.7594	0.5001	0.7593	0.5041	0.7614
QS	0.4931	0.7451	0.4885	0.7444	0.4986	0.7593
Post-retrieval Predictors						
CS	0.4858	0.7401	0.5051	0.7592	0.5014	0.7586
WIG	0.4858	0.7401	0.4964	0.7555	0.4959	0.7567
QF	0.5000	0.7548	0.5050	0.7607	0.5013	0.7560
Learning to Select						
KNN-Rel	0.5074	0.7533	0.5052 $\uparrow$	0.7607	0.5010	0.7612
KNN-KL	0.5191 $\uparrow$	0.7656 $\uparrow$	0.5120 $\uparrow$	0.7629 $\uparrow$	**0.5164** $\uparrow$ $*$	**0.7628** $\uparrow$
KNN-JS	0.5191 $\uparrow$	0.7656 $\uparrow$	0.5118 $\uparrow$	0.7623 $\uparrow$	**0.5164** $\uparrow$ $*$	**0.7628** $\uparrow$
Kmeans-Rel	0.5057	0.7610	0.5027	0.7590	0.4954	0.7477
Kmeans-KL	**0.5206** $\uparrow$	**0.7678** $\uparrow$	**0.5134** $\uparrow$	**0.7647** $\uparrow$	0.5138 $\uparrow$ $*$	0.7604
Kmeans-JS	**0.5206** $\uparrow$	**0.7678** $\uparrow$	0.5132 $\uparrow$	0.7631 $\uparrow$	0.5136 $\uparrow$ $*$	0.7614

Table 6.15: Evaluation of the selective application of collection enrichment on the TREC 2007 enterprise document search task using the PL2F weighting scheme.

that we use the k-means algorithm for identifying similar queries based on the query feature of the mean of the relevance score, the KL divergence score and the JS divergence score, respectively.

From the tables, we observe that the best retrieval performance (highlighted in bold) on each external resource is always achieved by our proposed learning to select framework, e.g., using the PL2F weighting scheme and on the TREC 2007 dataset (Table 6.15), the highest MAP and nDCG scores are: 0.5206 and 0.7678 on Wikipedia; 0.5134 and 0.7647 on Aquaint2; and 0.5164 and 0.7628 on .GOV. In particular, the best retrieval performance obtained by using our proposed learning to select framework constantly makes improvement over the document

TREC 2007 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
BM25F	0.4587 α	0.7166 α	0.4587 α	0.7166 α	0.4587 α	0.7166 α
+QE	0.4847 β	0.7495	0.4847 β	0.7495	0.4847 β	0.7495
+CE	0.4755 γ	0.7308	0.4646 γ	0.7232 γ	0.4656 γ	0.7226 γ
Pre-retrieval Predictors						
AvICTF	0.4809	0.7399	0.4735	0.7324	0.4727	0.7426
AvIDF	0.4791	0.7379	0.4773	0.7382	0.4798	0.7477
Gamma1	0.4897	0.7560	0.4857	0.7481	0.4841	0.7468
Gamma2	<u>0.4940</u>	<u>0.7590</u>	0.4895	0.7501	0.4813	0.7472
AvPMI	0.4759	0.7414	0.4811	0.7477	0.4716	0.7407
QS	0.4744	0.7333	0.4696	0.7332	0.4806	0.7484
Post-retrieval Predictors						
CS	0.4821	0.7421	0.4787	0.7395	<u>0.4915</u>	<u>0.7540</u>
WIG	0.4842	0.7413	<u>0.4915</u>	<u>0.7518</u>	0.4837	0.7363
QF	0.4815	0.7497	0.4825	0.7467	0.4821	0.7472
Learning to Select						
KNN-Rel	0.4916	0.7466	0.4980 $\uparrow$	0.7547 $\uparrow$	0.4951 $\uparrow$	0.7540
KNN-KL	**0.5063** $\uparrow$ *	**0.7640** $\uparrow$	**0.4999** $\uparrow$	0.7581 $\uparrow$	0.5000 $\uparrow$	0.7520
KNN-JS	**0.5063** $\uparrow$ *	**0.7640** $\uparrow$	**0.4999** $\uparrow$	0.7581 $\uparrow$	0.5001 $\uparrow$	0.7541 $\uparrow$
Kmeans-Rel	0.4817	0.7332	0.4837	0.7360	0.4804	0.7315
Kmeans-KL	0.5053 $\uparrow$	0.7623 $\uparrow$	0.4987 $\uparrow$	**0.7594** $\uparrow$	0.5021 $\uparrow$	0.7544 $\uparrow$
Kmeans-JS	0.5058 $\uparrow$	0.7607 $\uparrow$	0.4987 $\uparrow$	0.7584 $\uparrow$	**0.5035** $\uparrow$	**0.7562** $\uparrow$

Table 6.16: Evaluation of the selective application of collection enrichment on the TREC 2007 enterprise document search task using the BM25F weighting scheme.

weighting model, and the systematic application of QE/CE in both document weighting schemes across two different topic datasets and three different external resources in terms of MAP and nDCG. For example, on the TREC 2007 dataset and by using the Wikipedia external resource using the PL2F weighting scheme (Table 6.15), the obtained retrieval performances in MAP and nDCG are increased from: 0.4588 $\rightarrow$ 0.5206 and 0.7161 $\rightarrow$ 0.7678; 0.5015 $\rightarrow$ 0.5206 and 0.7596 $\rightarrow$ 0.7678; and 0.4859 $\rightarrow$ 0.5206 and 0.7349 $\rightarrow$ 0.7678. Moreover, such improvements are statistically significant in most cases (56 out of 72).

For the query performance predictor-based approach, marked improvement over the systematic application of query expansion is also observed across different external resources when an appropriate query performance predictor is used. In particular, the $\gamma 1$ (Equation (2.33)) and $\gamma 2$ (Equation (2.34)) predictors consistently boost the retrieval performance across three different external resources using the PL2F weighting scheme on the TREC 2008 dataset. In comparison with the best retrieval performance that is obtained by using the query performance predictor-based approach (emphasised with underline), we note that our proposed learning to select framework produces higher a retrieval performance in most cases. For example, on the TREC 2007 dataset and in the PL2F weighting scheme (Table 6.15), the retrieval performance is increased from 0.5113 $\rightarrow$ 0.5191 in MAP and 0.7654 $\rightarrow$ 0.7656 in nDCG when we use KNN-KL. Moreover, such improvements are statistically significant in some cases (marked with $*$), e.g., on the TREC 2007 dataset and by using the PL2F weighting scheme on the .GOV external resource: 0.5041 $\rightarrow$ 0.5164 in MAP.

Among three different query features, namely the mean of the relevance scores, the KL divergence score and the JS divergene score, we observe that the KL divergence score and the JS divergence score produce very close retrieval performance, which is explained in that they are mathematically related. In addition, both of them consistently outperform the mean of the relevance scores in both KNN and Kmeans algorithms. This observation is in line with what we found in Sections 6.4.3.3, 6.2.3.2, and 6.2.3.3. For example, in the PL2F weighting scheme and by using Wikipedia as the an external resource on the TREC 2007 dataset: 0.5074 $\rightarrow$ 0.5191 (KNN-Rel vs. KNN-KL) and 0.5074 $\rightarrow$ 0.5206 (Kmeans-Rel vs. Kmeans-KL) in MAP. With respect to the two different neighbouring query finding algorithms (KNN and k-means), both algorithms have shown their effectiveness and the obtained retrieval performances are quite close.

In order to better understand the accuracy of the two different selective application techniques, we have plotted their accuracy in the Receiver Operating Characteristics (ROC) space based on different external resources for the TREC 2007 and TREC 2008 datasets using different weighting schemes, respectively, as shown in Figures 6.1, 6.2, 6.3 and 6.4. Y and X axes provide True Positive Rate (TPR) and False Positive Rate (FPR), respectively. The TPR determines a classifier or a diagnostic test performance on classifying positive instances (CE in this case) correctly among all positive samples available during the test. FPR,

TREC 2008 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
PL2F	0.3564 α	0.5370 α	0.3564 α	0.5370 α	0.3564 α	0.5370 α
+QE	0.3695 β	0.5563	0.3695 β	0.5506	0.3695 β	0.5506
+CE	0.3629 γ	0.5506 γ	0.3504 γ	0.5378 γ	0.3544 γ	0.5478
Pre-retrieval Predictors						
AvICTF	0.3695	0.5495	0.3654	0.5454	0.3698	0.5448
AvIDF	0.3678	0.5474	0.3635	0.5453	0.3705	0.5488
Gamma1	0.3819	0.5571	0.3806	0.5610	0.3785	0.5602
Gamma2	0.3791	0.5567	0.3700	0.5558	0.3728	0.5546
AvPMI	0.3654	0.5506	0.3616	0.5498	0.3631	0.5458
QS	0.3661	0.5499	0.3557	0.5390	0.3580	0.5396
Post-retrieval Predictors						
CS	0.3730	0.5586	0.3666	0.5431	0.3675	0.5590
WIG	0.3715	0.5515	0.3672	0.5425	0.3706	0.5496
QF	0.3652	0.5492	0.3729	0.5466	0.3611	0.5434
Learning to Select						
KNN-Rel	0.3889 $\uparrow$	0.5683 $\uparrow$	0.3840 $\uparrow$	0.5541	0.3852 $\uparrow$	0.5575
KNN-KL	0.4025 $\uparrow *$	0.5765 $\uparrow$	**0.3945** $\uparrow *$	0.5587	**0.3930** $\uparrow *$	0.5608 $\uparrow$
KNN-JS	**0.4044** $\uparrow *$	**0.5767** $\uparrow$	**0.3945** $\uparrow *$	**0.5640** $\uparrow$	**0.3930** $\uparrow *$	**0.5618** $\uparrow$
Kmeans-Rel	0.3789	0.5559	0.3791	0.5561	0.3731	0.5507
Kmeans-KL	0.4013 $\uparrow *$	0.5725 $\uparrow$	0.3930 $\uparrow *$	0.5544	0.3913 $\uparrow *$	0.5572
Kmeans-JS	0.4009 $\uparrow$	0.5699 $\uparrow$	0.3930 $\uparrow *$	0.5538	0.3895 $\uparrow *$	0.5605 $\uparrow$

Table 6.17: Evaluation of the selective application of collection enrichment on the TREC 2008 enterprise document search task using the PL2F weighting scheme.

on the other hand, defines how many incorrect positive results occur among all negative samples (QE in this case) available during the test. The dashed line represents a random guess. For points above the dashed line, the further the distance from the line, the higher the accuracy.

From the figures, we notice that the accuracy of the query performance predictor-based approach is highly dependent on the selected weighting scheme and the used external resource. For example, on the TREC 2007 dataset and by using the .GOV collection as the external resource, the retrieval performance obtained by the AvICTF predictor is markedly above the dashed line in the PL2F weighting scheme (Figure 6.1). However, in the BM25F weighting scheme, the accuracy achieved by the same predictor is below the randome guess (Figure 6.2).

TREC 2008 Enterprise Document Search Task						
	Wikipedia		Aquaint2		.GOV	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
BM25F	0.3366 α	0.5220 α	0.3366 α	0.5220	0.3366 α	0.5220
+QE	0.3423 β	0.5233 β	0.3423 β	0.5229	0.3423 β	0.5229
+CE	0.3382 γ	0.5229 γ	0.3225 γ	0.5205	0.3325 γ	0.5178
Pre-retrieval Predictors						
AvICTF	0.3362	0.5188	0.3343	0.5246	0.3490	0.5286
AvIDF	0.3372	0.5182	0.3354	0.5276	0.3476	0.5245
Gamma1	0.3385	0.5169	0.3396	0.5252	0.3326	0.5158
Gamma2	0.3355	0.5156	0.3378	0.5250	0.3338	0.5200
AvPMI	0.3336	0.5196	0.3307	0.5166	0.3459	0.5256
QS	0.3359	0.5186	0.3284	0.5264	0.3427	0.5219
Post-retrieval Predictors						
CS	0.3538	0.5337	0.3341	0.5141	0.3424	0.5228
WIG	0.3495	0.5301	0.3357	0.5217	0.3453	0.5237
QF	0.3420	0.5216	0.3437	0.5245	0.3395	0.5226
Learning to Select						
KNN-Rel	0.3539 ↑	0.5223	0.3523 ↑ ∗	0.5237	0.3575 ↑	0.5278
KNN-KL	**0.3702** ↑ ∗	0.5376 ↑	0.3578 ↑ ∗	0.5260	**0.3644** ↑ ∗	0.5289 ↑
KNN-JS	0.3696 ↑ ∗	**0.5414** ↑ ∗	**0.3579** ↑ ∗	0.5252	**0.3644** ↑ ∗	**0.5290** ↑
Kmeans-Rel	0.3496	0.5333	0.3387	0.5174	0.3415	0.5226
Kmeans-KL	0.3688 ↑ ∗	0.5356 ↑	0.3546 ↑ ∗	0.5145	**0.3644** ↑ ∗	0.5199
Kmeans-JS	0.3680 ↑ ∗	0.5347 ↑	0.3529 ↑ ∗	**0.5287** ↑	0.3623 ↑ ∗	0.5268

Table 6.18: Evaluation of the selective application of collection enrichment on the TREC 2008 enterprise document search task using the BM25F weighting scheme.

Moreover, we can see that most predictors are not always above the dashed line. For example, the AvIDF predictor points are constantly above the dashed line on the TREC 2007 dataset through all external resources in the PL2F weighting scheme (see Figure 6.1). However, it always performs close or worse than a random guess on all external resources for the TREC 2008 dataset (see Figure 6.3). Furthermore, the accuracy of some predictors is close to a random guess which suggests that the selective application of collection enrichment for enterprise document search is a challenging task, e.g., the accuracy of the CS predictor on the TREC 2007 dataset using the BM25F weighting scheme is close to the dashed line on both the Wikipedia and Aquaint2 collections.

Among nine different predictors, only the $\gamma 1$ predictor (Equation (2.33)) has a constantly marked improvement over a random guess in the PL2F weighting scheme. In addition, we also observe that the accuracy of our proposed learning to select framework can consistently and markedly improve over the random guess for all external resources on both datasets. The only exception is when the mean of the relevances is used as a query feature on the TREC 2007 dataset and by using the PL2F weighting scheme on the .GOV external resource.

The above observations suggest that our proposed learning to select framework is effective in selectively applying an appropriate collection for expanding the initial query. In addition, our proposed use of divergence as a query feature is more robust than the mean of the relevances.

6.4.3.3 Importance of the External Resource

The performance of collection enrichment is dependent on the usefulness of the expansion terms extracted from the pseudo-relevant set, which is typically obtained from the top ranked documents on the external resource. In this part, we investigate the importance of the external resource for the selective application of collection enrichment.

From Tables 6.13 and 6.14, we note that the systematic application of CE can decrease the retrieval performance over the retrieval performance that is obtained by using the document weighting model only. For example, in the PL2F document weighting scheme and for the MAP evaluation measure, the retrieval performance decreases from 0.3564 to 0.3504 on the TREC 2008 dataset after applying CE on the Aquaint2 collection for all queries. However, systematically applying CE on the Wikipedia collection can always enhance the retrieval performance for both the TREC 2007 and TREC 2008 datasets using both the PL2F and the BM25F weighting scheme.

Moreover, from Tables 6.15, 6.16, 6.17 and 6.18, we observe that the performance of the query performance predictor-based approach is dependent on the selected external resource, e.g., in the BM25F weighting scheme, the $\gamma2$ predictor (Equation (2.34)) makes an improvement on the TREC 2007 dataset when the external resource is Wikipedia or Aquaint2, whereas, the retrieval performance decreases when the .GOV collection is used as the external resource. This is probably because the topics covered in the .GOV collection are more concentrated on the government domain, while the Wikipedia and Aquaint2 collections comprise

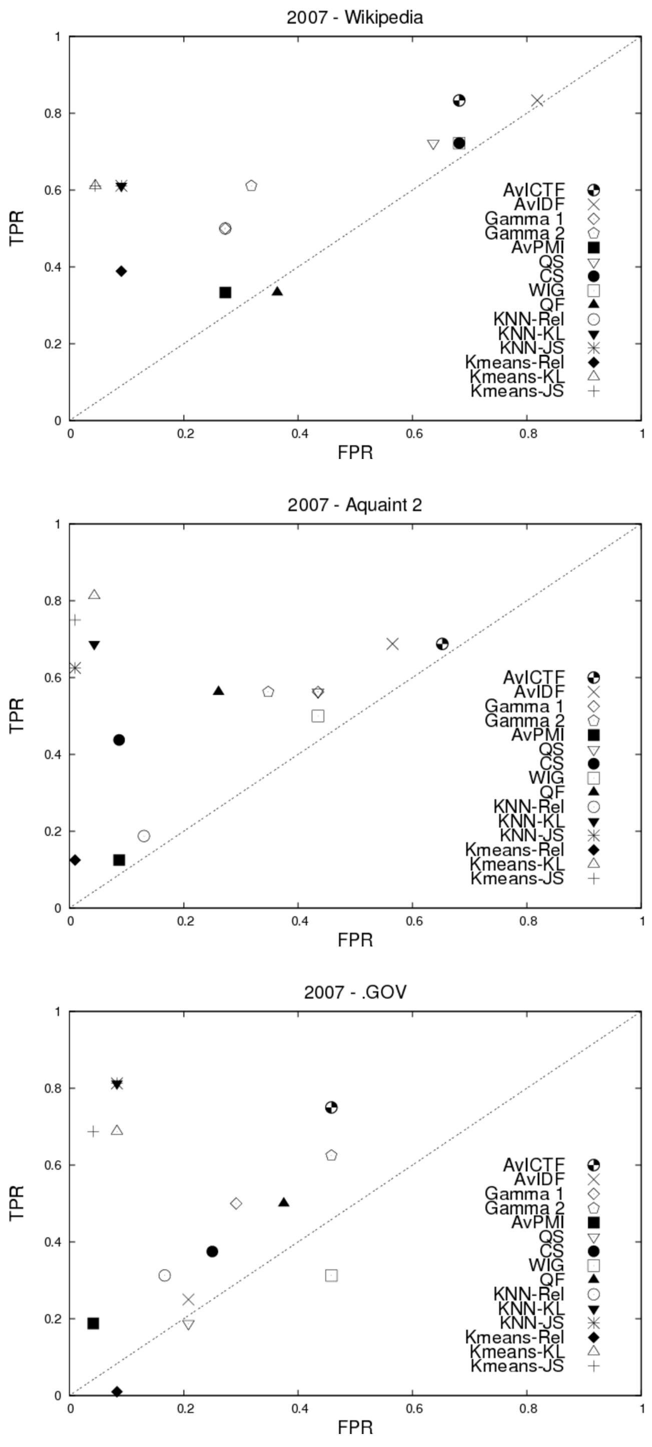


Figure 6.1: Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2007 enterprise document search using the PL2F weighting scheme.

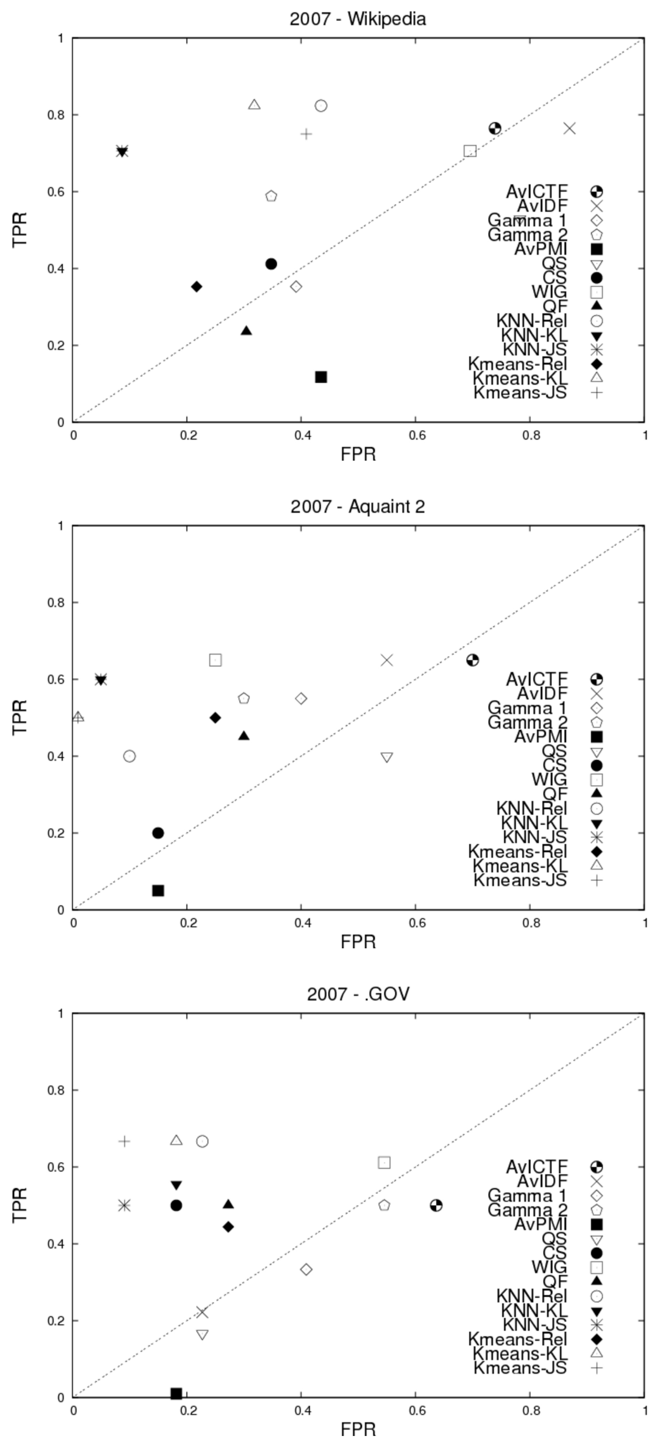


Figure 6.2: Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2007 enterprise document search using the BM25F weighting scheme.

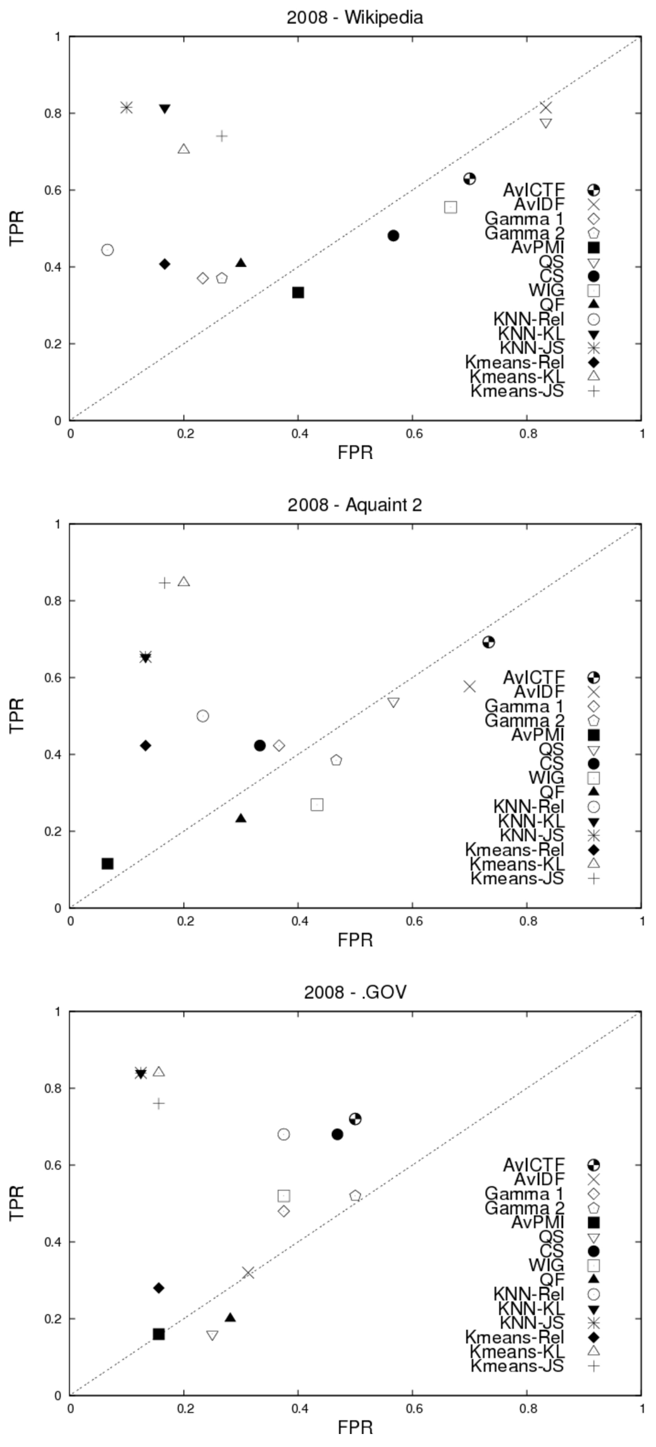
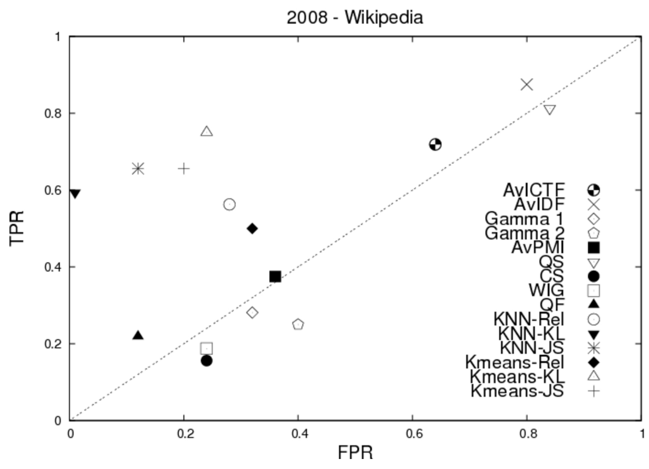
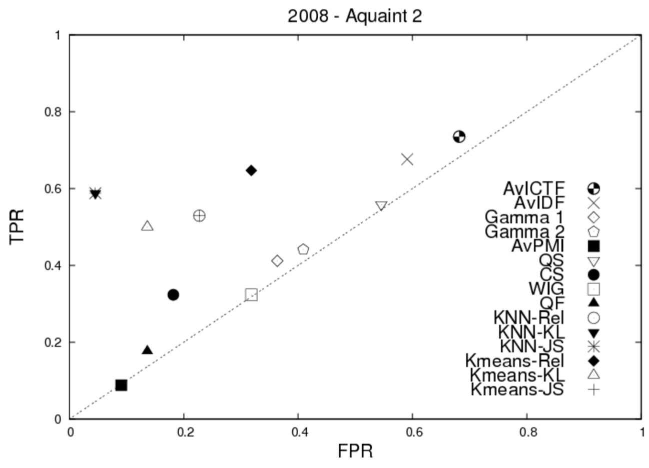


Figure 6.3: Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2008 enterprise document search using the PL2F weighting scheme.





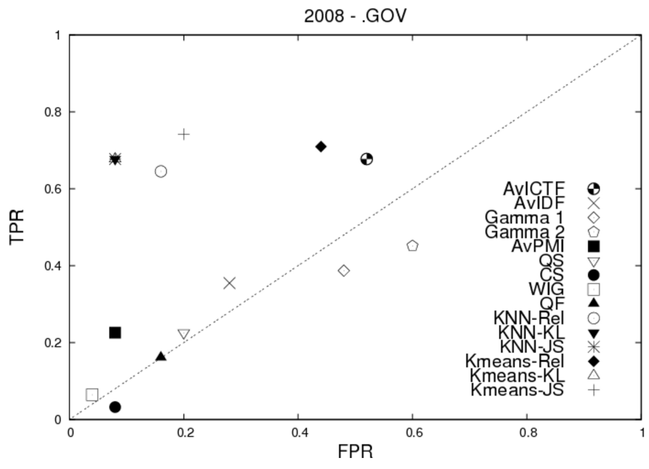


Figure 6.4: Comparison between different selective application techniques by plotting ROC space based on different external resources for TREC 2008 enterprise document search using the BM25F weighting scheme.

a broader range of topics, such as technology, science, etc. In addition, we also observe that the proposed learning to select framework is not highly dependent on the external resource chosen, as the retrieval performance can be consistently improved on all external resources by using either the KL divergence score or the JS divergence score as the query feature. In particular, the best retrieval performances for both TREC 2007 and TREC 2008 enterprise document search tasks are obtained from the Wikipedia collection, e.g. 0.5206 in MAP on TREC 2007 (see Table 6.15) and 0.4044 in MAP on TREC 2008 (see Table 6.17).

The above observations confirm that choosing an appropriate external resource before applying the selective collection enrichment is important and that the Wikipedia collection is the most appropriate one for the studied enterprise document search.

6.4.4 Summary

In this section, we have deployed the learning to select framework for selectively applying an appropriate resource for expanding a given query for the enterprise document search task. In addition, a full study of the effectiveness of our proposed learning to select framework in this application has been investigated on the TREC Enterprise CERC test collection and its corresponding 2007 and 2008 topic sets, in combination with three different external resources and several different baselines, including the query performance predictor-based approach.

Our experimental results showed that the retrieval performance for enterprise search can be significantly enhanced with the optimal selective application of CE (Section 6.4.3.1). By comparing the learning to select framework with several different baselines, we found that our proposed approach is effective for the selective application of collection enrichment. In addition, we also observed the robustness of our proposed approach as it always enhances retrieval performance on all external resources by using our proposed query features (Section 6.4.3.2).

In addition, by investigating the importance of different external resources for selective collection enrichment, we found that Wikipedia is the most useful external resource for enterprise search among those tested (Section 6.4.3.3). This is probably because most 2007 and 2008 enterprise topics are science-oriented, which are better covered in the Wikipedia collection.

6.5 Summary

In this chapter, we explored the deployment of the learning to select framework in various search applications, including selectively integrating an appropriate query independent feature into a document weighting scheme (Section 6.2), selective estimation of the relative importance of different query aspects in a search diversification task (Section 6.3), and selective application of an appropriate resource for expanding a given query (Section 6.4). The obtained experimental results from these search applications showed that our proposed learning to select framework can consistently make improvements on the retrieval performance, in comparison with these systematic application approaches. In particular, such improvements are significant in some cases. Moreover, the learning to select framework can also outperform these existing selective retrieval approaches, such as the query performance predictor-based approach (Section 3.4). This suggests that learning to select is an effective framework for selective application and emphasises the generality of the learning to select framework.

The investigations of the components of the learning to select framework showed that our proposed use of divergence measures as query features for identifying neighbouring queries was always more effective than the mean of the relevance scores, this observation is in line with the conclusions found in Chapter 5. In addition, for the neighbouring query finding component, both KNN and k-means have shown their effectiveness across these search applications, and the retrieval performances obtained by these two techniques are quite close, which is also in consistency with the observation in Chapter 5

Chapter 7

Conclusions and Future Work

7.1 Contributions and Conclusions

This thesis has investigated the selective application of an appropriate ranking function for Information Retrieval (IR) on a per-query basis, regardless of the query type and the number of candidate ranking functions/document features. This section discusses the contributions and conclusions of this thesis.

7.1.1 Contributions

The main contributions of this thesis are as follows:

- A novel framework, called learning to select (LTS), has been proposed for selectively applying an appropriate ranking function on a per-query basis, regardless of the given query's type and the number of candidate ranking functions/document features (Section 4.2). In the learning to select framework, the effectiveness of a ranking function for an unseen query is estimated based on its performance on the similar (i.e. neighbouring) already seen queries. In addition, the main components of the learning to select framework have been defined: query features (Section 4.4.1) and identification of neighbouring queries (Section 4.4.2).

- In the proposed learning to select framework, a classification algorithm is modified to identify neighbouring queries by using a query feature (Section 4.4.2). In particular, we propose to use the divergence measure to build such a query feature (Section 4.4.1). Two divergence measures have been used as query features, namely the Kullback-Leibler (KL) and the Jensen-Shannon (JS) divergence measures. These divergence measures quantify the extent to which a document ranking function alters the scores of an initial ranking of documents.

- The differences between the learning to select framework and the existing selective retrieval approaches have been discussed (Section 4.6). The existing selective retrieval approaches include query dependent ranking (Section 3.3), selective collection enrichment (Section 3.4), selective Web IR (Section 3.5), and query type prediction (Section 3.6). Overall, compared to these existing selective retrieval approaches, our proposed learning to select framework represents a general approach for selectively applying an appropriate ranking function on a per-query basis, which is agnostic to the number of candidate ranking functions/document features and the type of the queries.

- In Chapter 5, the proposed LTS framework has been thoroughly evaluated on two large standard document feature sets, which contain as many as 64 document features, and their corresponding TREC tasks. The features cover a wide range of query dependent and query independent features, such as the frequency of query terms in a document and PageRank (Equation (2.69)). In particular, several high-performance baselines are used in the evaluation, including a simulated query type prediction approach with a 100% accuracy in predicting the type of a given query and three state-of-the-art Learning to Rank (LTR) techniques, namely Ranking SVM (Section 3.2.2.3), AdaRank (Section 3.2.3.1), and the AFS method (Section 3.2.3.1). In particular, two different tasks have been investigated: selecting an appropriate ranking function from a number of candidate ranking functions (Section 5.5), and selecting a set of appropriate document features from a number of candidate document features for building an effective ranking function (Section 5.6). Moreover, to test the robustness of the learning to select framework, the selective application of an appropriate ranking function has been conducted by varying the number of candidate ranking functions from 2 to 63 (Section 5.5.2).

- To show the generality of the learning to select framework, we have deployed it for selectively integrating an appropriate query independent feature into a document weighting scheme (Section 6.2). This allows the document features to be applied to the queries that are more likely to benefit from these features. Moreover, a full study of the effectiveness of the learning to select framework in this search application has been investigated on the TREC 2003 and TREC 2004 Web track datasets. In particular, a simulated oracle query type prediction approach is used as one of our baselines (see Section 6.2.3.3).

- We have also deployed the learning to select framework for the selective estimation of the relative importance of different query aspects in a search diversification task (Section 6.3). This allows the ranked list of documents to provide a complete coverage of different interpretations for an ambiguous query. In addition, the effectiveness of the learning to select framework in this search application has been thoroughly tested using the large-scale TREC ClueWeb09 (category B) test collection, which contains over 50 million documents, and its corresponding TREC 2009 Web track topics set. Moreover, to test the robustness of the learning to select framework, three different weighting schemes, including BM25 (Equation (2.6)), DPH (Equation (2.21)), and language modelling (Equation (2.12)) were used.

- Furthermore, the learning to select framework has been deployed for the choice of an appropriate resource (e.g. an external document collection, such as Wikipedia) for expanding and enriching an initial query, for document search within an enterprise (Section 6.4). This alleviates the mismatch problem between query terms and the intranet documents. As mentioned in Section 6.4, the mismatch problem is severe in an enterprise, due to the sparsity of the vocabulary used within an enterprise. In addition, a full study of the effectiveness of the learning to select framework in this search application has been investigated on the TREC Enterprise CERC test collection and its corresponding 2007 and 2008 topic sets, in combination with three different external resources (see Section 6.4.2). Moreover, the query performance predictor-based approach is used as one of the baselines. In particular, nine different predictors (Section 2.4.3), including both pre-retrieval and post-retrieval predictors, were used in this investigation.

7.1.2 Conclusions

This section discusses the achievements and conclusions of this work.

Effectiveness of Learning to Select The evaluation of the effectiveness of the proposed learning to select framework in Chapter 5 includes two main investigations: selecting an appropriate ranking function from a number of candidate ranking functions (Section 5.5), and selecting a set of appropriate document features from a number of candidate document features for building a ranking function (Section 5.6).

In the first investigation, we compared the retrieval performance obtained by the learning to select framework with a simulated query type prediction approach that has a 100% accuracy in predicting the type of a given query, as well as three state-of-the-art Learning To Rank (LTR) techniques, namely Ranking SVM (Section 3.2.2.3), AdaRank (Section 3.2.3.1), and the AFS method (Section 3.2.3.1). Our experimental results showed that the retrieval performance obtained by using our proposed learning to select framework could consistently outperform the query type prediction approach and three state-of-the-art learning to rank techniques in terms of the MAP measure on different datasets (see Section 5.5.1). In addition, improvements over the query type prediction approach and all learning to rank techniques were statistically significant in most cases (see Tables 5.4, 5.5, 5.6 and 5.7 in Section 5.5.1).

The experimental results in the second investigation showed that our proposed framework is also effective for selecting multiple appropriate document features for building a ranking function and the obtained retrieval performance can be further enhanced when we increase the number of candidate document features from 2 to 6 (see Tables 5.8, 5.9, 5.10 and 5.11 in Section 5.6). These results attest to the effectiveness of the learning to select framework.

Robustness of Learning to Select We investigated the robustness of our learning to select framework by increasing the number of candidate ranking functions from 2 to 63. To achieve this, we simulate a number of candidate ranking functions by applying a learning to rank technique on several different combinations of document features (see Section 5.5.2).

By plotting the distribution of MAP versus the number of candidate ranking functions (see Section 5.5.2), we found that by using the learning to select framework, the obtained retrieval performance can be enhanced when increasing the number of candidate ranking functions. This suggests that learning to select is a robust framework for selectively applying an appropriate ranking function and emphasises the point that the learning to select framework is agnostic to the number of candidate ranking functions used.

Generality of Learning to Select To show the generality of the learning to select framework, we explored the deployment of the learning to select framework in various search applications, ranging from the selective application of query independent features in the Web search task (Section 6.2), to selectively applying a sub-query importance estimator in the search diversification task (Section 6.3), or selective application of query expansion for the enterprise document search task (Section 6.4).

The obtained experimental results from these search applications showed that our proposed learning to select framework can consistently make improvements on the retrieval performance, in comparison with several strong baselines (see Sections 6.2.3.3, 6.3.4.2, and 6.4.3.2). In particular, such improvements are significant in some cases (e.g. see Table 6.6 in Section 6.2.3.3). Moreover, the learning to select framework can also outperform the existing selective retrieval approaches, such as the query performance predictor-based approach (Section 3.4). These results attest to the generality of the learning to select framework.

The Components of Learning to Select Furthermore, to better understand the proposed learning to select framework, the components (i.e., the neighbouring query finding technique and query features) of the learning to select framework have been thoroughly investigated. Two different neighbouring query finding techniques have been described in Section 4.4.2, namely KNN and k-means. Three different query features have been introduced in Section 4.4.1, namely, the mean of the relevance scores, the KL and the JS divergence measures.

Experimental results from Chapter 5, Sections 6.2, 6.3 and 6.4 have shown that our proposed use of divergence measures (i.e. KL and JS) as query features to identify neighbouring queries was always more effective than the mean of the relevance scores measure, which ignores the distribution of relevance scores. Besides, for the neighbouring query finding component, both KNN and k-means have shown their effectiveness across all search applications (e.g. Table 6.3 in Section 6.2.3.2). In addition, based on the three used query features, the retrieval performances obtained by these two identification of neighbouring queries techniques are quite close (e.g. Table 5.5 in Section 5.5).

7.2 Directions for Future Work

This section discusses several directions for future work related to, or stemming from this thesis.

Other Query Features Three different query features (i.e., the KL divergence measure, the JS divergence measure and the mean of the relevance scores) have been evaluated in the proposed learning to select framework in this thesis. In our evaluation results, both divergence measures (i.e. KL and JS) outperform the mean of the relevance scores. The divergence measures are used to determine the extent to which a document ranking function alters the scores of an initial ranking of documents. Apart from the divergence measure, the correlation measure (e.g., Pearson's correlation coefficient (Pearson, 1896)) could be used as a query feature. The correlation measures the dependence between two variables. In our case, the two variables correspond to a base ranking function and a candidate ranking function (see Section 4.4.1). Hence, the correlation measure could be used to determine the strength of the dependence between the two variables.

Query log is a record of user interactions with search engines, such as the user queries, the number of corresponding document clicks, and the URL of the clicked documents (Dupret et al., 2006). Several different query log-based features have been extracted and used for different search applications (Baeza-Yates & Tiberi, 2007; Dupret & Mendoza, 2006; Joachims, 2002; Lee et al., 2005; Liu et al., 2006).

For example, Lee et al. (2005) proposed the use of the number of clicks to detect the user intent, as the informational queries usually have higher number of clicks than the navigational queries. These query log-based feature could be used as query features for identifying similar queries. For example, similar queries might have similar clicked documents. However, this requires the availability of large query logs, which are currently not available in academia.

Multi-Dimensional Query Feature Space The experiments conducted in this thesis used a single query feature (e.g., a divergence measure or the mean of the relevance scores) to identify neighbouring queries. Several other possible query features have been discussed above. Similar to the growing trend of building a ranking function from a large number of document features (see Section 3.2), we plan to build a multi-dimensional query feature space with the aim of achieving a higher retrieval performance. In addition, techniques that can be used to combine document features could also be deployed for combining the query features, such as the linear combination (Berry et al., 1995) and the FLOE method (described in Section 2.6.4).

Identifying Neighbouring Queries In this thesis, we used two different classification-based techniques (i.e. KNN and k-means (see Section 4.4.2)) to identify the neighbouring queries for a given test query. In the future, we will investigate some other neighbouring queries identification approaches. For example, Baeza-Yates & Tiberi (2007) generated the semantic relationships between queries by representing queries in a vector space based on the query-click bipartite graph. Inspired by this work, we plan to employ the semantic relationships of queries, as derived from query logs, to identify neighbouring queries.

Learning to Select in Other Applications In this thesis, we have investigated the effectiveness of the learning to select framework in various search applications. In the future, we plan to deploy the learning to select framework in other fields, such as the blogosphere. With the advent of Web 2.0, more and more Web users report their daily life, as well as publish their views and opinions on various events as they happen using blogging tools (Ounis et al., 2006). There are two main search tasks based on the analysis of a commercial blog search engine: opinion-finding (i.e., “What do people think about X?”) and blog distillation (i.e., “Find me a blog with a principal, recurring interest in X.”) (Ounis et al., 2006).

The opinion-finding task requires finding not only relevant, but also opinionated blog posts for a given topic. Hannah et al. (2007) proposed two different approaches for this task: the first one is a light-weight dictionary-based statistical approach and the second one applies techniques in Natural Language Processing (NLP) for subjectivity analysis. These two different approaches lead to different retrieval performances on the TREC Blog track dataset. By treating the ranking functions generated by the these two different opinion-finding techniques as candidate ranking functions, the proposed learning to select framework could be used to selectively apply the most appropriate technique on a per-query basis, so as to enhance the opinion-finding performance. Moreover, the number of candidate ranking functions can be extended by adding some other opinion-finding techniques, such as the use of document priors (e.g. the number of comments for each post) for finding opinionated blog post (Ernsting et al., 2007).

For the blog distillation search task, various approaches have been developed (Ounis et al., 2008), such as the use of expert search techniques to rank blogs, the use of folksonomies to expand the queries and the use of the query likelihood language modelling approach. By treating the ranking functions created by these techniques as candidate ranking functions, our proposed learning to select framework could also be used to selectively apply an appropriate blog retrieval technique on per-query basis for the blog distillation search task.

Appendix A

Additional Tables

ID	Feature Description
1	Term frequency (TF) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF * IDF of body
12	TF * IDF of anchor
13	TF * IDF of title
14	TF * IDF of URL
15	TF * IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor

ID	Feature Description
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document
36	LMIR.JM of body
37	LMIR.JM of anchor
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink base score propagation: weighted in-link
44	Hyperlink base score propagation: weighted out-link
45	Hyperlink base score propagation: uniform out-link
46	Hyperlink base feature propagation: weighted in-link
47	Hyperlink base feature propagation: weighted out-link
48	Hyperlink base feature propagation: uniform out-link
49	HITS authority
50	HITS hub
51	PageRank
52	HostRank
53	Topical PageRank
54	Topical HITS authority
55	Topical HITS hub
56	Inlink number
57	Outlink number
58	Number of slash in URL
59	Length of URL
60	Number of child page
61	BM25 of extracted title
62	LMIR.ABS of extracted title
63	LMIR.DIR of extracted title
64	LMIR.JM of extracted title

Table A.1: Document features included in the LETOR 3.0 dataset.

ID	Feature Description
1	Term frequency (TF) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF * IDF of body
12	TF * IDF of anchor
13	TF * IDF of title
14	TF * IDF of URL
15	TF * IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document
36	LMIR.JM of body
37	LMIR.JM of anchor
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of whole document
41	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page

Table A.2: Document features included in the LETOR 4.0 dataset.

Bibliography

Agrawal, R., Gollapudi, S., Halverson, A. & Ieong, S. (2009). Diversifying search results. *in* 'WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining'. ACM. New York, NY, USA. pp. 5–14. 2.7, 6.3.3

Allan, J., Ballesteros, L., Callan, J. P., Croft, W. B. & Lu, Z. (1995). Recent experiments with inquery. *in* 'In The Fourth Text REtrieval Conference (TREC-4)'. pp. 49–63. 2.4.3.1

Allan, J., Carterette, B., Aslam, J. A., Pavlu, V., Dachev, B. & Kanoulas, E. (2007). Million query track 2007 overview. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 16th Text Retrieval Conference (TREC 2007)'. National Institute of Standards and Technology (NIST). 5.3.1

Amati, G. (2003). *Probability Models for Information Retrieval based on Divergence From Randomness*. Ph.D. Thesis. University of Glasgow, UK. 2.3.3, 2.3.3.2, 2.3.3.3, 2.3.3.3, 2.4.1.1, 2.4.1.1, 2.4.1.1, 2.5.2, 6.4.2

Amati, G., Ambrosi, E., Bianchi, M., Gaibisso, C. & Gambosi, G. (2007). Fub, iasi-cnr and university of tor vergata at trec 2007 blog track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 16th Text Retrieval Conference (TREC 2007)'. National Institute of Standards and Technology (NIST). 2.3.3.4

Amati, G., Carpineto, C. & Romano, G. (2004). Query difficulty, robustness, and selective application of query expansion. *Advances in Information Retrieval* pp. 127–137. 2.4.3

Arampatzis, A., van der Weide, T., Koster, C. H. A. & van Bommel, P. (2000). An evaluation of linguistically-motivated indexing schemes. *in* 'In Proceedings of the 22nd BCS-IRSG Colloquium on IR Research'. 2.5

Attar, R. & Fraenkel, A. S. (1977). Local feedback in full-text retrieval systems. *J. ACM* **24**(3), 397–417. 2.4.1

Austen, J. (1813). *Pride and Prejudice*. T. Egerton, Whitehall. UK. 2.2

Baeza-Yates, R. (2003). Information retrieval in the web: beyond current search engines. *International Journal of Approximate Reasoning* **34**, 97–104. 2.1

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley. 2.1, 2.2.2, 2.2.3, 2.2.4

Baeza-Yates, R. & Tiberi, A. (2007). Extracting semantic relations from query logs. *in* 'KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining'. ACM. New York, NY, USA. pp. 76–85. 7.2

Baeza-Yates, R., Hurtado, C. & Mendoza, M. (2005). Query recommendation using query logs in search engines. pp. 588–596.

Bailey, P., Craswell, N., de Vries, A. P. & Soboroff, I. (2007). Overview of the trec 2007 enterprise track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 16th Text Retrieval Conference (TREC 2007)'. National Institute of Standards and Technology (NIST). 2.4.2, 6.4.2

Balog, K., Bron, M., He, J., Hofmann, K., Meij, E., de Rijke, M., Tsagkias, M. & Weerkamp, W. (2009). The university of amsterdam at trec 2009. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 18th Text Retrieval Conference (TREC 2009)'. National Institute of Standards and Technology (NIST). 6.3

Balog, K., Hofmann, K., Weerkamp, W. & de Rijke, M. (2007). Query and document models for enterprise search. *in* 'The Sixteenth Text Retrieval Conference (TREC 2007)'. NIST. Special Publication. 6.4

Berger, A. L., Pietra, V. J. D. & Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.* **22**(1), 39–71. 3.2.1.2

Berry, M. W., Dumais, S. T. & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review* **37**(4), 573–595. 7.2

Blair, D. C. (2002). Some thoughts on the reported results of trec. *Inf. Process. Manage.* **38**(3), 445–451. 2.7

Breese, J. S., Heckerman, D. & Kadie, C. M. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *in* G. F. Cooper & S. Moral, eds, 'UAI'. Morgan Kaufmann. pp. 43–52. 3.2.2.2

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* **30**(1–7), 107–117. 1.1, 2.6, 2.6.1, 2.6.1

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum* **36**(2), 3–10. 1.1

Buckley, C. & Voorhees, E. M. (2004). Retrieval evaluation with incomplete information. *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 25–32. 2.7

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **2**(2), 121–167. 3.2.1.2, 3.2.1.2

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. & Hullender, G. (2005). Learning to rank using gradient descent. *in* 'ICML '05: Proceedings of the 22nd international conference on Machine learning'. ACM. New York, NY, USA. pp. 89–96. 1.1, 3.2.2, 3.2.2.1, 3.2.2.1, 3.2.3.1, 3.2.3.1

Cai, D., He, X., Wen, J.-R. & Ma, W.-Y. (2004). Block-level link analysis. *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 440–447. 2.6

Callan, J. (2000). Distributed information retrieval. *in* 'In: Advances in Information Retrieval'. Kluwer Academic Publishers. pp. 127–150. 6.3.1

Cao, G., Nie, J.-Y., Gao, J. & Robertson, S. (2008). Selecting good expansion terms for pseudo-relevance feedback. *in* 'SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 243–250. 2.4.2, 3.2.2.3

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y. & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. *in* 'SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 186–193. 3.2.2, 3.2.2.3

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F. & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *in* 'ICML '07: Proceedings of the 24th international conference on Machine learning'. ACM. New York, NY, USA. pp. 129–136. 1.1, 3.1, 3.2.3, 3.2.3.2

Carbonell, J. & Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. *in* 'SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 335–336. 6.3.1

Carpineto, C. & Romano, G. (2004). *Concept Data Analysis: Theory and Applications*. John Wiley & Sons. 2.4

Carpineto, C., de Mori, R., Romano, G. & Bigi, B. (2001). An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.* **19**(1), 1–27. 2.4.1

Chakrabarti, S., Khanna, R., Sawant, U. & Bhattacharyya, C. (2008). Structured learning for non-smooth ranking losses. *in* 'KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining'. ACM. New York, NY, USA. pp. 88–96. 3.2.3

Chandar, P., Kailasam, A., Muppaneni, D., Thota, L. & Carterette, B. (2009). Ad hoc and diversity retrieval at the university of delaware. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 18th Text Retrieval Conference (TREC 2009)'. National Institute of Standards and Technology (NIST). 6.3

Clarke, C., Craswell, N. & Soboroff, I. (2004). Overview of the trec 2004 terabyte track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. National Institute of Standards and Technology (NIST). 2.7, 5.3.2

Clarke, C. L., Craswell, N. & Soboroff, I. (2009). Preliminary report on the trec 2009 web track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 18th Text Retrieval Conference (TREC 2009)'. National Institute of Standards and Technology (NIST). 2.7, 6.3

Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S. & MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation. *in* 'SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 659–666. 2.7, 6.3.3

Cleverdon, C. W. (1962). Aslib cranfield research project: report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. *Cranfield Library.* 2.7

Cohen, W. W., Schapire, R. E. & Singer, Y. (1998). Learning to order things. *in* 'NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10'. MIT Press. Cambridge, MA, USA. pp. 451–457. 3.2.2

Cook, R. D. & Weisberg, S. (1982). Criticism and influence analysis in regression. *Sociological Methodology* **13**, 313–361. 3.2.1.1

Cooper, W. S., Gey, F. C. & Dabney, D. P. (1992). Probabilistic retrieval based on staged logistic regression. *in* 'SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 198–210. 3.2.1.3

Cossock, D. & Zhang, T. (2006). Subset ranking using regression. *in* 'COLT'06: proceedings of the 19th Annual Conference on Learning Theory'. Pittsburgh, Pennsylvania, USA. pp. 605–619. 3.2.1.3, 4.6

Cover, T. M. & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13(1)**, 21–27. 1.3, 4.4, 4.4.2.1

Craswell, N. & Hawking, D. (2002). Overview of the trec 2002 web track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 11th Text Retrieval Conference (TREC 2002)'. National Institute of Standards and Technology (NIST). 5.3.1, 5.3.2, 6.4.2

Craswell, N. & Hawking, D. (2004). Overview of the trec 2004 web track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. National Institute of Standards and Technology (NIST). 1.2, 2.3.1, 3.6, 3.6, 3.7, 4.1

Craswell, N., Fetterly, D., Najork, M., Robertson, S. & Yilmaz, E. (2009). Microsoft research at trec 2009. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 18th Text Retrieval Conference (TREC 2009)'. National Institute of Standards and Technology (NIST). 6.3

Craswell, N., Robertson, S., Zaragoza, H. & Taylor, M. (2005). Relevance weighting for query independent evidence. *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 416–423. 1.1, 2.6, 2.6.3, 2.6.4, 2.6.4, 3.1, 6.2, 6.2.2

Craswell, N., Zaragoza, H. & Robertson, S. (2005). Microsoft cambridge at trec 14: Enterprise track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 14th Text Retrieval Conference (TREC 2005)'. National Institute of Standards and Technology (NIST).

Croft, W. B. & Harper, D. J. (1997). Using probabilistic models of document retrieval without relevance information. pp. 339–344. 2.3.1

Cronen-Townsend, S., Zhou, Y. & Croft, W. B. (2002). Predicting query performance. *in* 'SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 299–306. 2.4.3, 2.4.3.2, 3.4

Diaz, F. & Metzler, D. (2006). Improving the estimation of relevance models using large external corpora. *in* 'SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 154–161. 2.4.2, 2.4.3, 6.4

Donald, M., Croft, W. B. & Mccallum, A. (2005). Direct maximization of rank-based metrics for information retrieval. Technical report. University of Massachusetts. 4.6

Duda, R. O., Hart, P. E. & Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. 2 edn. Wiley-Interscience. 4.4.2

Duda, R. O., Hart, P. E. & Stork, D. G. (2001). *Pattern Classification*. 2. edn. Wiley. New York. 3.2.1.2

Dupret, G. & Mendoza, M. (2006). Automatic query recommendation using click-through data. *IFIP International Federation for Information Processing* **218**, 303–312. 7.2

Dupret, G., Piwowarski, B., Hurtado, C. & Mendoza, M. (2006). A statistical model of query log generation. *in* 'Proceedings of the 13th edition of the Symposium on String Processing and Information Retrieval (SPIRE 2006)'. pp. 217–228. 7.2

Elias, P. (1975). Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on* **21**(2), 194–203. 2.2.4

Ernsting, B., Weerkamp, W. & de Rijke, M. (2007). Language modeling approaches to blog post and feed finding. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 16th Text Retrieval Conference (TREC 2007)'. National Institute of Standards and Technology (NIST). 7.2

Fagan, J. L. (1987). Automatic phrase indexing for document retrieval: An examination of syntactic and non-syntactic methods. *in* ‘SIGIR’. ACM. pp. 91–101. 2.5

Fagin, R., Kumar, R., McCurley, K. S., Novak, J., Sivakumar, D., Tomlin, J. A. & Williamson, D. P. (2003). Searching the workplace web. *in* ‘WWW ’03: Proceedings of the 12th international conference on World Wide Web’. ACM. New York, NY, USA. pp. 366–375. 6.4

Feldman, S. & Sherman, C. (2003). The high cost of not finding information. *Technical Report*. 6.4

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications, Vol. 1 (Volume 1)*. 3 edn. Wiley. 2.3.3.2

Feustela, C. D. & Shapiro, L. G. (1982). The nearest neighbor problem in an abstract metric space. *Pattern Recognition Letters* **1**, 125–128. 4.4.2.1

Frakes, W. B. & Baeza-Yates, R. A., eds (1992). *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall. 2.2.2, 2.2.4

Freund, Y. (1990). Boosting a weak learning algorithm by majority. *in* ‘COLT ’90: Proceedings of the third annual workshop on Computational learning theory’. Morgan Kaufmann Publishers Inc.. San Francisco, CA, USA. pp. 202–216. 3.2.2.2

Freund, Y. & Schapire, R. (1999). A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.* **14**(5), 771–780. 3.2.2.2

Freund, Y. & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *in* ‘EuroCOLT ’95: Proceedings of the Second European Conference on Computational Learning Theory’. Springer-Verlag. London, UK. pp. 23–37. 3.2.3.1

Freund, Y., Iyer, R., Schapire, R. E. & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.* **4**, 933–969. 3.2.2, 3.2.2.2, 3.2.2.2, 3.2.2.2, 3.2.3.1

Fuhr, N. (1989). Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Trans. Inf. Syst.* **7**(3), 183–204. 3.2.1.1

Gao, J., Nie, J.-Y., Wu, G. & Cao, G. (2004). Dependence language model for information retrieval. *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 170–177. 2.5, 2.5

Geng, X., Liu, T.-Y., Qin, T., Arnold, A., Li, H. & Shum, H.-Y. (2008). Query dependent ranking using k-nearest neighbor. *in* 'SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 115–122. 1.1, 1.2, 3.1, 3.3, 3.3.2, 3.3.3, 3.3.3, 4.4, 4.4.1, 4.6

Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. *in* 'SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval'. Springer-Verlag New York, Inc.. New York, NY, USA. pp. 222–231. 3.2.1.3

Golomb, S. (1966). Run-length encodings (corresp.). *Information Theory, IEEE Transactions on* **12**(3), 399–401. 2.2.4

Hannah, D., Macdonald, C., Peng, J., He, B. & Ounis, I. (2007). University of glasgow at trec 2007: Experiments in blog and enterprise tracks with terrier. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 16th Text Retrieval Conference (TREC 2007)'. National Institute of Standards and Technology (NIST). 2.4.1.1, 6.2, 6.2.2, 7.2

Harman, D. (1993). Overview of the first trec conference. *in* 'SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 36–47. 2.7

Harris, L. E. (2006). Foundations of library and information science, 2nd edition: Book reviews. *J. Am. Soc. Inf. Sci. Technol.* **57**(9), 1279–1280. 2.6.1

Harter, S. P. (1975). An algorithm for probabilistic indexing. *Journal of the American Socity for Information Science* **26**(4), 280–289. 2.3.1

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc.. New York, NY, USA. 3.2.1.1

Hauff, C., Murdock, V. & Baeza-Yates, R. (2008). Improved query difficulty prediction for the web. *in* 'CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 439–448. 2.4.3, 2.4.3.1, 2.4.3.1, 3.4

He, B. & Ounis, I. (2006). Query performance prediction. *Information System* **31**(7), 585–594. 1.2, 2.4.3, 2.4.3.1, 2.4.3.1, 2.4.3.1

He, B., Peng, J. & Ounis, I. (2009). Fitting score distribution for blog opinion retrieval. *in* 'SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 688–689. 3.7, 4.1, 4.4.1

Herbrich, R., Graepel, T. & Obermayer, K. (2000). *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA. 1.1, 3.2, 3.2.2, 3.2.2.3, 5.3.3

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. *in* 'ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries'. Springer-Verlag. London, UK. pp. 569–584. 2.3.2

Hiemstra, D. (2001). *Using Language Models for Information Retrieval*. CTIT Ph.D. Thesis Series No. 01-32. Taaluitgeverij Neslia Paniculata. Enschede. 2.3.2

Hill, W., Stead, L., Rosenstein, M. & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. *in* 'CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems'. ACM Press/Addison-Wesley Publishing Co.. New York, NY, USA. pp. 194–201. 3.2.2.2

Jansen, B. J., Spink, A., Bateman, J. & Saracevic, T. (1998). Real life information retrieval: a study of user queries on the web. *SIGIR Forum* **32**(1), 5–17. 2.4

Järvelin, K. & Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. *in* 'SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 41–48. 4.6

Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.* **20**(4), 422–446. 2.7

Joachims, T. (2002). Optimizing search engines using clickthrough data. *in* 'KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining'. ACM. New York, NY, USA. pp. 133–142. 3.2, 3.2.2, 3.2.2.3, 3.2.2.3, 3.2.2.3, 3.2.3.1, 5.3.3, 7.2

Joachims, T. (2005). A support vector method for multivariate performance measures. *in* 'ICML '05: Proceedings of the 22nd international conference on Machine learning'. ACM. New York, NY, USA. pp. 377–384. 3.2.3.1

Kamps, J., Mishne, G. & de Rijke, M. (2004). Language models for searching in web corpora.. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. Vol. Special Publication 500-261. National Institute of Standards and Technology (NIST). 1.1, 1.2, 2.6, 6.2, 6.2.2

Kelly, D. & Belkin, N. J. (2001). Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback. *in* 'SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 408–409. 2.4.1

Kraaij, W., Westerveld, T. & Hiemstra, D. (2002). The importance of prior probabilities for entry page search. *in* 'SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 27–34. 1.1, 2.6, 2.6.2, 6.2

Kullback, S. (1997). *Information Theory and Statistics (Dover Books on Mathematics)*. Dover Publications. 1.3, 4.4.1

Kwok, K. L. (1996). A new method of weighting query terms for ad-hoc retrieval. *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 187–195. 2.4.3.1

Kwok, K. L. & Chan, M. (1998). Improving two-stage ad-hoc retrieval for short queries. *in* 'SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 250–256. 2.4.2, 2.4.3, 6.4

Lafferty, J., Sleator, D. & Temperley, D. (1992). Grammatical trigrams: A probabilistic model of link grammar. *in* 'In Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language'. pp. 89–97. 2.5

Lavrenko, V. & Croft, W. B. (2001). Relevance based language models. *in* 'SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 120–127. 2.4.1.1

Lee, J. H. (1997). Analyses of multiple evidence combination. *in* 'SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 267–276. 4.4.1

Lee, U., Liu, Z. & Cho, J. (2005). Automatic identification of user goals in web search. *in* 'WWW '05: Proceedings of the 14th international conference on World Wide Web'. ACM. New York, NY, USA. pp. 391–400. 7.2

Lin, J. (1991). Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on* **37**(1), 145–151. 1.3, 4.4.1

Lioma, C., Macdonald, C., Plachouras, V., Peng, J., He, B. & Ounis, I. (2006). University of glasgow at trec 2006: Experiments in terabyte and enterprise tracks with terrier. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 15th Text

Retrieval Conference (TREC 2006)'. National Institute of Standards and Technology (NIST). 6.2, 6.2.2

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Found. Trends Inf. Retr.* **3**(3), 225–331. 1.1, 3.2, 3.2.2.1, 3.2.3

Liu, T.-Y., Xu, J., Qin, T., Xiong, W. & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *in* 'LR4IR 2007, in conjunction with SIGIR 2007'. 3.1, 3.2, 5.3.1, 5.3.2, 5.3.4

Liu, Y., Zhang, M., Ru, L. & Ma, S. (2006). Automatic query type identification based on click through information. *in* 'First International Workshop on Adaptive Information Retrieval (AIR)'. pp. 593–600. 7.2

Lo, R. T.-w., He, B. & Ounis, I. (2005). Automatically building a stopword list for an information retrieval system. *in* 'Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR'05)'. Utrecht, Netherlands. 2.2.2

Losee, Jr., R. M. (1994). Term dependence: truncating the bahadur lazarsfeld expansion. *Inf. Process. Manage.* **30**(2), 293–303. 2.5, 2.5

Lovins, J. B. (1968). Development of a stemming algorithm.. *Mechanical Translation and Computational Linguistics* **11**, 22–31. 2.2.3

Luce, R. D. (1959). *Individual Choice Behavior*. Wiley, New York, US. 3.2.3.2

Luhn, H. (1957). A statistical approach to mechanized encoding and searching of literary information.. *IBM Journal of Research and Development 1(4)* pp. 309–317. 2.2.2

Macdonald, C., Hannah, D. & Ounis, I. (2008). High quality expertise evidence for expert search. *in* 'Proceedings of 30th European Conference on Information Retrieval (ECIR08)'. 6.2

Macdonald, C., Plachouras, V., Ben, H., Lioma, C. & Ounis, I. (2005). University of glasgow at webclef 2005: Experiments in per-field normalisation and language specific stemming. *in* 'Proceedings of the Cross Language Evaluation Forum (CLEF) 2005'. 2.3.3.4, 2.3.3.4

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *in* L. M. L. Cam & J. Neyman, eds, 'Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability'. Vol. 1. University of California Press. pp. 281–297. 4.4.2.2

Mallows, C. L. (1975). Non-null ranking models. *Biometrika* **44**, 114–130. 3.2.3.2

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. *in* 'COLING-02: proceedings of the 6th conference on Natural language learning'. Association for Computational Linguistics. Morristown, NJ, USA. pp. 1–7. 3.2.1.2

Manmatha, R., Rath, T. & Feng, F. (2001). Modeling score distributions for combining the outputs of search engines. *in* 'SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 267–275. 3.3.3, 3.7, 4.1, 4.4.1

Manning, C. D., Raghavan, P. & Schtze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. New York, NY, USA. 2.2.1, 2.2.2, 2.2.3, 2.7

McCreadie, R., Macdonald, C., Ounis, I., Peng, J. & Santos, R. L. (2009). University of glasgow at trec 2009: Experiments with terrier. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 18th Text Retrieval Conference (TREC 2009)'. National Institute of Standards and Technology (NIST). 2.4.1.1, 6.3

Metzler, D. (2007). Automatic feature selection in the markov random field model for information retrieval. *in* 'CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management'. ACM. New York, NY, USA. pp. 253–262. 1.1, 3.1, 3.2.3.1, 5.3.3

Metzler, D. (2008). *Markov random fields for information retrieval*. Ph.D. Thesis. University of Massachusetts, Amherst, US. 2.5.1

Metzler, D. & Croft, W. B. (2005). A markov random field model for term dependencies. *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 472–479. 2.5, 2.5.1, 2.5.1, 2.5.1, 2.5.1, 2.5.2

Metzler, D. & Croft, W. B. (2007). Linear feature-based models for information retrieval. *Inf. Retr.* **10**(3), 257–274. 3.2.3.1

Metzler, D. & Kanungo, T. (n.d.). Machine learned sentence selection strategies for query-biased summarization. *in* 'Proceedings of SIGIR Learning to Rank Workshop'. 3.2.2.3

Metzler, D., Strohman, T., Zhou, Y. & Croft, W. (2005). Indri at trec 2005: Terabyte track. *in* 'The 14th Text Retrieval Conference (TREC 2005)'. 2.6, 6.2, 6.2.2

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicography* **3**, 235–244. 2.4.1

Mishne, G. & de Rijke, M. (2005). Boosting web retrieval through query operations. pp. 502–516. 2.5

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill. New York. 4.4.2.1

Mitra, M., Buckley, C., Singhal, A. & Cardie, C. (1997). An analysis of statistical and syntactic phrases.. *in* L. Devroye & C. Chrisment, eds, 'RIAO'. pp. 200–217. 2.5

Nallapati, R. (2004). Discriminative models for information retrieval. *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 64–71. 1.1, 3.2.1.2, 3.2.2.3

Nallapati, R. & Allan, J. (2002). Capturing term dependencies using a language model based on sentence trees. *in* 'CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 383–390. 2.5

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C. & Lioma, C. (2006). Terrier: A high performance and scalable information retrieval platform. *in* 'Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)'. 6.2.2, 6.3.3, 6.4.2

Ounis, I., de Rijke, M., Macdonald, C., Mishne, G. & Soboroff, I. (2006). Overview of the trec 2006 blog track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 15th Text Retrieval Conference (TREC 2006)'. National Institute of Standards and Technology (NIST). 7.2

Ounis, I., Macdonald, C. & Soboroff, I. (2008). Overview of the trec 2008 blog track. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 17th Text Retrieval Conference (TREC 2008)'. National Institute of Standards and Technology (NIST). 7.2

Paice, C. D. (1990). Another stemmer. *SIGIR Forum* **24**(3), 56–61. 2.2.3

Pearson, K. (1896). Mathematical contributions to the theory of evolution. iii. regression, heredity and panmixia. *Philos. Trans. Royal Soc. London Ser. A* **187**, 253–318. 7.2

Peng, J. & Ounis, I. (2007). Combination of document priors in web information retrieval. *in* 'Proceedings of the 29th European Conference on IR Research (ECIR 2007)'. Lecture Notes in Computer Science. Springer. 1.5

Peng, J. & Ounis, I. (2009). Selective application of query-independent features in web information retrieval. *in* 'ECIR '09: Proceedings of the 31st European Conference on IR Research on Advances in Information Retrieval'. Springer-Verlag. Berlin, Heidelberg. pp. 375–387. 1.1, 1.2, 1.5, 3.3.3, 3.6, 3.7, 4.1, 4.4.2, 4.4.2.3, 6.2.2

Peng, J., He, B. & Ounis, I. (2009). Predicting the usefulness of collection enrichment for enterprise search. *in* 'ICTIR '09: Proceedings of the 2nd International Conference on Theory of Information Retrieval'. Springer-Verlag. Berlin, Heidelberg. pp. 366–370. 1.1, 1.2, 1.5, 2.4.2, 3.1, 3.4, 6.4

Peng, J., Macdonald, C. & Ounis, I. (2008). Automatic document prior feature selection for web retrieval. *in* 'SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 761–762. 1.5

Peng, J., Macdonald, C. & Ounis, I. (2010). Learning to select a ranking function. *in* 'ECIR '10: Proceedings of the 32nd European Conference on IR Research on Advances in Information Retrieval'. Springer-Verlag. Berlin, Heidelberg. pp. 114–126. 1.5, 3.2.4, 1

Peng, J., Macdonald, C., He, B. & Ounis, I. (2007). Combination of document priors in web information retrieval. *in* 'RIAO 2007'. 1.5, 6.2, 6.2.2

Peng, J., Macdonald, C., He, B. & Ounis, I. (2009). A study of selective collection enrichment for enterprise search. *in* 'CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 1999–2002. 1.5

Peng, J., Macdonald, C., He, B., Plachouras, V. & Ounis, I. (2007). Incorporating term dependency in the dfr framework. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 843–844. 1.1, 1.5, 2.5, 2.5.2, 2.5.2, 2.5.2, 6.2.2

Pickens, J. & Croft, W. B. (2000). An exploratory analysis of phrases in text retrieval. *in* 'In Proceedings of RIAO (Recherche dInformation assiste par Ordinateur'. pp. 1179–1195. 2.5

Pietra, S. D., Pietra, V. J. D., Gillet, J., Lafferty, J. D., Printz, H. & Ures, L. (1994). Inference and estimation of a long-range trigram model. *in* 'ICGI '94: Proceedings of the Second International Colloquium on Grammatical Inference and Applications'. Springer-Verlag. London, UK. pp. 78–92. 2.5

Pirkola, A. & Järvelin, K. (2001). Employing the resolution power of search keys. *J. Am. Soc. Inf. Sci. Technol.* **52**(7), 575–583. 2.4.3.1

Plachouras, V. (2006). *Selective Web Information Retrieval*. Ph.D. Thesis. University of Glasgow, UK. 1.1, 1.2, 3.1, 3.5, 3.5

Plachouras, V. & Ounis, I. (2004). Usefulness of hyperlink structure for query-biased topic distillation. *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 448–455. 1.1, 1.2, 3.1

Plachouras, V., Cacheda, F., Ounis, I. & van Rijsbergen, C. J. (2003). University of glasgow at the web track: Dynamic application of hyperlink analysis using the query scope. *in* 'NIST Special Publication: SP 500-255 The Twelfth Text Retrieval Conference (TREC 2003)'. 2.4.3.1

Plachouras, V., He, B. & Ounis, I. (2004). University of glasgow at trec 2004: Experiments in web, robust and terabyte tracks with terrier. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. National Institute of Standards and Technology (NIST). 3.6

Ponte, J. M. & Croft, W. B. (1998). A language modeling approach to information retrieval. *in* 'SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 275–281. 2.3.2, 2.3.2, 2.3.2, 2.5.1

Porter, M. F. (1980). An algorithm for suffix stripping. *Program* **14**(3), 130–137. 2.2.3

Qin, T., Zhang, X.-D., Wang, D.-S., Liu, T.-Y., Lai, W. & Li, H. (2007). Ranking with multiple hyperplanes. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 279–286. 3.2.2

Robertson, S. E. (1977). The probability ranking principle in ir. *Journal of Documentation* **33**, 294–304. 2.3

Robertson, S. E. (1990). On term selection for query expansion. *J. Doc.* **46**(4), 359–364. 2.4.1, 2.4.1.1

Robertson, S. E. & Spärck Jones, K. (1988). Relevance weighting of search terms. pp. 143–160. 2.3, 2.3.1, 3.2.1.2

Robertson, S. E. & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *in* 'SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval'. Springer-Verlag New York, Inc.. New York, NY, USA. pp. 232–241. 2.3.1

Robertson, S. E., van Rijsbergen, C. J. & Porter, M. F. (1980). Probabilistic models of indexing and searching. *in* 'SIGIR '80: Proceedings of the 3rd annual ACM conference on Research and development in information retrieval'. Butterworth & Co.. Kent, UK, UK. pp. 35–56. 2.3.1

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M. & Gatford, M. (1994). Okapi at trec-3. *in* 'In The Third Text REtrieval Conference (TREC-3)'. pp. 109–126. 1.1, 2.3.1

Robertson, S. E., Zaragoza, H. & Taylor, M. (2004). Simple bm25 extension to multiple weighted fields. *in* 'CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 42–49. 2.3.1

Rocchio, J. (1971*a*). *Relevance Feedback in Information Retrieval.* pp. 313–323. 2.4.1

Rocchio, J. (1971*b*). Relevance feedback in information retrieval export find similar. *In The SMART Retrieval System* pp. 313–323. 2.4.3

Salton, G. & McGill, M. J. (1986*a*). *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc.. New York, NY, USA. 2.3

Salton, G. & McGill, M. J. (1986*b*). *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc.. New York, NY, USA. 2.3

Salton, G., Fox, E. A. & Wu, H. (1983*a*). Extended boolean information retrieval. *Commun. ACM* **26**(11), 1022–1036. 2.3

Salton, G., Fox, E. A. & Wu, H. (1983*b*). Extended boolean information retrieval. *Commun. ACM* **26**(11), 1022–1036. 2.4.3

Santos, R. L. T., Peng, J., Macdonald, C. & Ounis, I. (2010). Explicit search result diversification through sub-queries. *in* 'ECIR '10: Proceedings of the 32nd European Conference on IR Research on Advances in Information Retrieval'. Springer-Verlag. Berlin, Heidelberg. 6.3

Santos, R. L. T., Macdonald, C. & Ounis, I. (2010). Exploiting query reformulations for web search result diversification. *in* '19th International Conference on World Wide Web (WWW 2010)'. 6.3.1, 6.3.1

Scholer, F., Williams, H. E., Yiannis, J. & Zobel, J. (2002). Compression of inverted indexes for fast query evaluation.. *in* 'SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. pp. 222–229. 2.2.4

Schütze, H. & Pedersen, J. O. (1997). A cooccurrence-based thesaurus and two applications to information retrieval. *Inf. Process. Manage.* **33**(3), 307–318. 2.4.1

Serdyukov, P., Aly, R. & Hiemstra, D. (2008). University of twente at the trec 2008 enterprise track: Using the global web as an expertise evidence source. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 17th Text Retrieval Conference (TREC 2008)'. National Institute of Standards and Technology (NIST). 6.2

Shokouhi, M. (2007). Central-rank-based collection selection in uncooperative distributed information retrieval. *in* G. Amati, C. Carpineto, G. Romano, G. Amati, C. Carpineto & G. Romano, eds, 'ECIR'. Vol. 4425 of *Lecture Notes in Computer Science*. Springer. pp. 160–172. 6.3.1

Silverstein, C., Marais, H., Henzinger, M. & Moricz, M. (1999). Analysis of a very large web search engine query log. *SIGIR Forum* **33**(1), 6–12. 2.4

Singhal, A., Buckley, C. & Mitra, M. (1996). Pivoted document length normalization. *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 21–29. 2.3.3

Skiścim, C. C. & Golden, B. L. (1983). Optimization by simulated annealing: A preliminary computational study for the tsp. *in* 'WSC '83: Proceedings of the 15th conference on Winter Simulation'. IEEE Press. Piscataway, NJ, USA. pp. 523–535. 5.3.4, 6.2.2, 6.3.3, 6.4.2

Song, F. & Croft, W. B. (1999). A general language model for information retrieval. *in* 'CIKM '99: Proceedings of the eighth international conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 316–321. 2.5.1

Song, R., Wen, J.-R., Shi, S., Xin, G., Liu, T.-Y., Qin, T., Zheng, X., Zhang, J., Xue, G.-R. & Ma, W.-Y. (2004). Microsoft research asia at web track and terabyte track of trec 2004.. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. Vol. Special Publication 500-261. National Institute of Standards and Technology (NIST). 1.2, 3.1, 3.6

Spärck Jones, K. (1971). *Automatic Keyword Classification for Information Retrieval*. Butterworth, London, UK. 2.4.1

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* **28**, 11–21. 2.3

Spärck Jones, K. & van Rijsbergen, C. (1976). Information retrieval test collections. *Journal of documentation* **32**, 59–75. 2.7, 5.3.1

Spärck Jones, K. & Willett, P., eds (1997). *Readings in information retrieval*. Morgan Kaufmann Publishers Inc.. San Francisco, CA, USA. 2.3

Spärck Jones, K., Robertson, S. E. & Sanderson, M. (2007). Ambiguous requests: implications for retrieval tests, systems and theories. *SIGIR Forum* **41**(2), 8–17. 6.3

Srikanth, M. & Srihari, R. (2002). Biterm language models for document retrieval. *in* 'SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 425–426. 2.5, 2.5.1

Taylor, M., Guiver, J., Robertson, S. & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. *in* 'WSDM '08: Proceedings of the international conference on Web search and web data mining'. ACM. New York, NY, USA. pp. 77–86. 3.1, 3.2.3, 3.2.3.1, 3.2.3.1

Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H. & Ma, W.-Y. (2007). Frank: a ranking method with fidelity loss. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 383–390. 3.2.2

Upstill, T., Craswell, N. & Hawking, D. (2003). Query-independent evidence in home page finding. *ACM Trans. Inf. Syst.* **21**(3), 286–313. 1.1

van Rijsbergen, C. J. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation* **33**(2), 106–199. 2.5

van Rijsbergen, C. J. (1979). *Information Retrieval.* 2nd edition edn. Butterworth-Heinemann. London. 2.1

Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. *in* 'SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval'. Springer-Verlag New York, Inc.. New York, NY, USA. pp. 61–69. 2.4.1

Voorhees, E. M. (2001). The philosophy of information retrieval evaluation. *in* 'CLEF '01: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems'. Springer-Verlag. London, UK. pp. 355–370. 2.7

Voorhees, E. M. (2008). Common evaluation measures. *in* 'The 17th Text Retrieval Conference (TREC 2008)'. New York, NY, USA. 2.7

Voorhees, E. M. & Harman, D. (2000). Overview of the sixth text retrieval conference (trec-6). *Inf. Process. Manage.* **36**(1), 3–35. 2.7, 5.3.1

Wang, C., Jing, F., Zhang, L. & Zhang, H.-J. (2007). Learning query-biased web page summarization. *in* 'CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management'. ACM. New York, NY, USA. pp. 555–562. 3.2.2.3

White, R. W., Jose, J. M. & Ruthven, I. (2001). Comparing explicit and implicit feedback techniques for web retrieval: Trec-10 interactive track report. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 10th Text Retrieval Conference (TREC 2001)'. National Institute of Standards and Technology (NIST). 2.4.1

Williams, H. E. & Zobel, J. (1999). Compressing integers for fast file access. *Computer Journal* **42**(3), 193–201. 2.2.4

Witten, I. H., Moffat, A. & Bell, T. C. (1999). Managing gigabytes: Compressing and indexing documents and images. Morgan Kaufmann. San Francisco. 2.2.4

Xia, F., Liu, T.-Y., Wang, J., Zhang, W. & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. *in* 'ICML '08: Proceedings of the 25th international conference on Machine learning'. ACM. New York, NY, USA. pp. 1192–1199. 3.2.3

Xu, J. & Croft, W. B. (1996). Query expansion using local and global document analysis. *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 4–11. 2.4.1

Xu, J. & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 391–398. 1.1, 3.1, 3.2, 3.2.3, 3.2.3.1, 5.3.3

Xue, G.-R., Yang, Q., Zeng, H.-J., Yu, Y. & Chen, Z. (2005). Exploiting the hierarchical structure for link analysis. *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 186–193. 5.3.2

Yang, K., Yu, N., Wead, A., Rowe, G. L., Li, Y., Friend, C. & Lee, Y. (2004). Widit in trec 2004 genomics, hard, robust and web tracks. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. National Institute of Standards and Technology (NIST). 1.2, 3.1, 3.6

Yilmaz, E. & Aslam, J. A. (2006). Estimating average precision with incomplete and imperfect judgments. *in* 'CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management'. ACM. New York, NY, USA. pp. 102–111. 2.7

Yom-Tov, E., Fine, S., Carmel, D. & Darlow, A. (2005). Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 512–519. 2.4.3

Yue, Y., Finley, T., Radlinski, F. & Joachims, T. (2007). A support vector method for optimizing average precision. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 271–278. 3.2.3

Zaragoza, H., Craswell, N., Taylor, M., Saria, S. & Robertson, S. (2004). Microsoft cambridge at trec 13: Web and hard tracks. *in* E. M. Voorhees & L. P. Buckland, eds, 'The 13th Text Retrieval Conference (TREC 2004)'. National Institute of Standards and Technology (NIST). 2.3.1

Zhai, C. & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.* **22**(2), 179–214. 2.3.2

Zhou, Y. & Croft, W. B. (2007). Query performance prediction in web search environments. *in* 'SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval'. ACM. New York, NY, USA. pp. 543–550. 2.4.3, 2.4.3.2, 2.4.3.2

Zhu, X. & Gauch, S. (2000). Incorporating quality metrics in centralized/distributed information retrieval on the world wide web. *in* ‘SIGIR ’00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval’. ACM. New York, NY, USA. pp. 288–295. 2.6.3

Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments?. *in* ‘SIGIR ’98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval’. ACM. New York, NY, USA. pp. 307–314. 2.7