MODELING REFORMULATION AS QUERY DISTRIBUTIONS

A Dissertation Presented

by

XIAOBING XUE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

July 17, 2012

Computer Science

MODELING REFORMULATION AS QUERY DISTRIBUTIONS

A Dissertation Presented

by

XIAOBING XUE

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

David A. Smith, Member

Weibo Gong, Member

Lori A. Clarke, Department Chair
Computer Science

To my parents and Xuan.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support that I received from many people. First and foremost, I would like to thank my advisor, W. Bruce Croft, for his guidance and supervision during my Ph.D. study. Bruce is an excellent advisor, who provided many insightful suggestions and comments for my research. He taught me to think deeply and focus on the fundamental problems. He also encouraged me to explore different areas of Information Retrieval, which helps me form a broad background.

I would also like to thank my thesis committee members, James Allan, David A. Smith and Weibo Gong for their valuable comments on my thesis.

I should also thank all members of Center for Intelligent Information Retrieval (CIIR). I thank R. Manmatha for his suggestions. I thank Jiwoon Jeon, Giridhar Kumaran, Jinyoung Kim, Van Dang and Samuel Huston for their collaborations. I thank Michael Bendersky, Xing Yi, Jangwon Seo, Jeffery Dalton, Elif Aktolga, Marc Cartright and Henry Field for their discussions. I thank David Fisher, Kate Moruzzi, Dan Parker, Jean C. Joyce and Glenn Stowell for their supports.

I should also acknowledge Xiaoxin Yin, Hang Li and Daxin Jiang from Microsoft Research for their mentoring during my summer internships. I have learned a lot from them about how to do good research in an industrial environment.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advance Research Projects Agency (DARPA) under contract number HR0011-06-C-0023, in part by NSF grant #IIS-0711348, and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommenda-

tions expressed in this material are those of the author and do not necessarily reflect those of the sponsor.

ABSTRACT

MODELING REFORMULATION AS QUERY DISTRIBUTIONS

July 17, 2012

XIAOBING XUE

B.Sci., NANJING UNIVERSITY, NANJING, CHINA

M.Eng., NANJING UNIVERSITY, NANJING, CHINA

Directed by: Professor W. Bruce Croft

Query reformulation modifies the original query with aim of providing a better representation of a user's information need and consequently improving the retrieval performance. Previous reformulation models typically generate words and phrases related to the original query, but do not consider how these words and phrases would fit together in realistic or actual queries. Some recent work on web search studies specific reformulation operations, but ignores how to combine different operations within the same framework. Furthermore, little research considers the reformulation model and the retrieval model from a joint perspective.

In this dissertation, a novel framework is proposed that models reformulation as a distribution of reformulated queries, where each reformulated query is associated with a probability indicating its importance. On one hand, this framework considers a reformulated query as the basic unit and can capture the important query-level dependencies between words and phrases in a realistic or actual query. On the other

hand, since a reformulated query is the output of applying a single or multiple reformulation operations, this framework combines different operations such as query segmentation, query substitution and query deletion within the same framework. Moreover, a retrieval model is considered as an integrated part of this framework, which considers the reformulation model and the retrieval model jointly.

Specifically, the query distribution framework consists of three major components, which are query generation, probability estimation and retrieval. For query generation, we generate the reformulated queries that are semantically related to the original query using different operations. For probability estimation, we estimate the probability assigned to each reformulated query by directly optimizing the retrieval performance. For retrieval, the retrieval scores from each reformulated query are combined together and the probabilities are used as the combination weights.

Furthermore, in order to model the relationships between the reformulated queries, we extend the standard query distribution model to the hierarchical query distribution. The hierarchial query distribution model transforms the original query into a reformulation tree, where each path of the tree models a sequence of generating reformulated queries. A stage-based probability estimation approach is proposed to capture the relationships between queries and directly optimize the retrieval performance.

Several implementations of the query distribution model are designed for different types of queries and applications including short keyword queries, verbose queries, natural language questions and patent applications. Experiments on TREC collections show that the query distribution model significantly and consistently outperforms the state-of-the-art techniques.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xiii**

LIST OF FIGURES . . . . . **xvi**

CHAPTER

1. INTRODUCTION **1**

1.1 Query Reformulation in Information Retrieval 2
1.2 Query Reformulation in Web Search 4
1.3 Interactive vs. Automatic 5
1.4 Motivations 6
1.5 A Query Distribution Model 8
1.6 Contributions 9
1.7 Organization 9

2. RELATED WORK **11**

2.1 Query Reformulation in Information Retrieval 11
2.2 Query Reformulation in Web Search 14
2.3 Distribution-based Query Representation 17
2.4 Combing Reformulation Model and Retrieval Model 17
2.5 Modeling the relationships between reformulated queries 18

3. FRAMEWORK OF QUERY DISTRIBUTIONS **20**

3.1 Framework 20
3.2 Comparison with Previous Reformulation Models 23

3.2.1 Concept Distribution (CDist) 23
3.2.2 Single Reformulated Query (SRQ) 25

3.2.3 Query Generation Process . . . 27
3.2.4 Retrieval Scores . . . 29

3.3 Summary . . . 31

4. REFORMULATED QUERY GENERATION . . . 33

4.1 Passage Analysis for Keyword Queries . . . 33

4.1.1 Generating Query Substitutions . . . 34
4.1.2 Generating Query Segmentations . . . 37
4.1.3 Examples . . . 39

4.2 Subset Selection for Verbose Queries . . . 40
4.3 Q&A Search for Natural Language Questions . . . 41

4.3.1 Retrieval Model for Q&A Archives . . . 42
4.3.2 Learning Word-to-Word Translation Probabilities . . . 44
4.3.3 Examples . . . 47

4.4 Patent Transformation For Prior-art Queries . . . 48

4.4.1 Operational Queries Generation . . . 49
4.4.2 Examples . . . 51

4.5 Summary . . . 52

5. PROBABILITY ESTIMATION . . . 53

5.1 Framework . . . 53
5.2 Optimizing the performance of using a single reformulated query . . . 55
5.3 Optimizing the performance of using a set of reformulated queries . . . 58
5.4 Summary . . . 61

6. APPLICATION I: QUERY DISTRIBUTION FOR SHORT QUERIES . . . 62

6.1 Implementation of Query Distribution Model . . . 62

6.1.1 Query Features . . . 63
6.1.2 Retrieval Model . . . 64

6.2 Experimental Configuration . . . 68
6.3 Examples . . . 70
6.4 Results . . . 72
6.5 Summary . . . 79

7. APPLICATION II: QUERY DISTRIBUTION FOR VERBOSE QUERIES 80

7.1 Implementation of Query Distribution Model 81

7.1.1 Query Features 81
7.1.2 Retrieval Model 83

7.2 Experimental Configuration 86
7.3 Examples 87
7.4 Results 90
7.5 Summary 91

8. HIERARCHICAL QUERY DISTRIBUTION 94

8.1 Framework 96
8.2 Hierarchical Query Distribution for Verbose Queries 97

8.2.1 Building Tree Structure 98
8.2.2 Weight Estimation 99
8.2.3 Query Features 102
8.2.4 Retrieval Model 103

8.3 Experimental Configuration 103

8.3.1 Example 105
8.3.2 Retrieval Performance 105
8.3.3 Further Analysis 108
8.3.4 Subset Selection vs. Query Substitution 111
8.3.5 Parameter Analysis 111

8.4 Summary 112

9. CONCLUSION AND FUTURE WORK 115

9.1 Summary 115
9.2 Contributions 117
9.3 Future Work 118

APPENDICES

A. BACKGROUND 121

A.1 Retrieval Models 121
A.2 Indri Query Language 122

A.3 Collections 123
A.4 Evaluation Metrics 125

B. PROOFS 127

Bibliography 129

BIBLIOGRAPHY 129

LIST OF TABLES

Table **Page**

3.1 The query distribution representation for "oil industry history" 21

3.2 Different query representations for the original query "oil industry history" 25

4.1 Examples of reformulated queries generated using the passage analysis technique 39

4.2 Examples of the subset queries selected from a verbose query 41

4.3 Word-to-word translation probability examples. Each column shows the top 10 target terms for a given source term. TTable denotes the type of the word-to-word translation probability table. 47

4.4 Examples of the reformulated questions. 48

4.5 Possible values of the parameters 51

4.6 Examples of Search Queries 51

6.1 Implementation of the Query Distribution Model for short queries. 63

6.2 Three types of features for the reformulated query q_r 64

6.3 The Indri queries for "(petroleum industry)(history)" 66

6.4 Query distributions learned for the original query "oil industry history" 67

6.5 Summary of Baselines 70

6.6 Example of the query distribution learned on the non-stemmed index using the document-level retrieval model. Top ranked reformulated queries (q_r) are displayed. In the query distribution, the original query (q) is italicized. Average Precision (AP) is reported as the retrieval performance. 71

6.7 Example of the query distribution learned on the non-stemmed index using the document-level and the passage-level retrieval models. Top ranked reformulated queries are displayed. In the query distribution, the original query is italicized. Average Precision (AP) is reported as the retrieval performance. 73

6.8 The results of different reformulation models. The best performance is bolded. * denotes significantly different with baselines. 74

6.9 Further analysis of query distribution. QDist denotes QDist(doc). 76

6.10 The performance of top three queries in QDist(doc) using different types of features. p denotes PSG, n denotes NGRAM and o denotes OPER. 79

7.1 Implementation of the Query Distribution Model for verbose queries. 81

7.2 Three types of features 84

7.3 Example of Indri queries. 85

7.4 Summary of Baselines 87

7.5 Example of the subset distribution learned using SubQL. Top four subset queries are displayed. In the original query q, the words actually used are bolded and stopwords and stop structures are italicized. 88

7.6 Comparisons of the top subset queries in the subset distributions learned using different retrieval models. In the original query q, the words actually used are bolded and stopwords and stop structures are italicized. 89

7.7 Performance of retrieval models using the subset distribution. ^{q} denotes significantly different with QL and $_d$ denotes significantly different with SDM. 90

7.8 The mean average precision (MAP) of using different types of features. “I” denotes the Independency Features, “L” denotes the Local Dependency Features and “G” denotes the Global Dependency Features. 92

8.1 Summary of features 102

8.2 Summary of Baselines 104

8.3 Examples of the reformulation tree. The top ranked nodes are displayed. In the original query q, the stopwords and stop structures are italicized. 106

8.4 Comparisons of retrieval performance. * denotes significantly different from the baseline. 107

8.5 Comparisons with QL+SubQL and SDM+SubQL. "+", "=" and "-" denote that RTree performs better, equal or worse than QL+SubQL and SDM+SubQL with respect to MAP. 109

8.6 The effect of subset query selection and query substitution with respect to MAP 111

8.7 The effect of the parameter $ModNum$ with respect to MAP 112

A.1 TREC collections used in experiments 124

LIST OF FIGURES

Figure **Page**

3.1 The framework of the query distribution model. 21

3.2 The queries generated by different models 29

4.1 Algorithm for generating reformulated queries. 38

4.2 An example patent. 50

4.3 General algorithm for transforming the query patent to an effective search query. 50

5.1 The framework for probability estimation 54

5.2 The process of optimizing the performance of a single reformulated query. 58

5.3 The process of optimizing the performance of a set of reformulated queries 60

6.1 The effect of the number of reformulated queries. Non-stemmed index is used. x-axis is the number of top ranked reformulated queries and y-axis is MAP. 78

7.1 The influence of the number of subset queries 93

8.1 The reformulation sequences generated for the verbose query q "any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt" 94

8.2 From the query distribution to the hierarchical query distribution for the verbose query "identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt" 95

8.3 The process of constructing a reformulation tree 99

8.4 Analysis of relative increases/decreases of MAP over QL. 110

8.5 The effect of the parameter *SubNum*. x-axis denotes *SubNum* and y-axis denotes MAP. 113

A.1 Indri queries implementing the query likelihood language model (QL) and the sequential dependence model (SDM) 123

A.2 An example of TREC topic 124

CHAPTER 1

INTRODUCTION

The query is essential in information retrieval, since it represents a user's information need and is used as the input to a search system. The original query formulated by a user, however, is not always a good representation of a user's underlying information need, especially from a search system's perspective. First, a typical user query is short. For example, on average, a web search query consists of two or three keywords. Considering that only a small number of keywords are included in the original query, many important concepts can be missed. Given a TREC query[1] "oil industry history", the information need is about the history of the U.S. oil industry including historical exploration and drilling. Besides the query words, other words such as "energy", "production", "exploration" and "drilling" are also important for a search system to find relevant documents, but are not included in the original query. Second, due to the complexities of natural languages, the same information need can be expressed in different ways. For example, the query "oil industry history" can be also expressed as "petroleum industry history" or "oil and gas industry history". The search system only using the original query may miss relevant documents containing alternative expressions of the information need.

On the other hand, search queries have been evolving beyond short keyword queries. For example, verbose queries (Bendersky and Croft 2008; Kumaran and Carvalho 2009) that describe a user's information need in detail, have been studied in *ad hoc* retrieval and web search. Natural language questions are also

[1]The details of TREC queries will be described in the Appendices.

widely observed as search queries in question and answer archives (Jeon *et al.* 2005). For patent retrieval, the whole patent application (Xue and Croft 2009) is used as a query. From a user's perspective, these new types of queries help express the information needs naturally and save the effort of selecting keywords. However, for a search system, these types of queries are difficult to process and the additional information provided is more likely to confuse current search systems rather than help them (Kumaran and Carvalho 2009).

Therefore, in this dissertation, we study *Query Reformulation* as a process of modifying the original query posed by a user to provide a better representation of the underlying information need for a search system.

From a probabilistic view, the original query is just one possible output of a query generation process. This process generates queries according to a probabilistic distribution reflecting a user's underlying information need. Thus, query reformulation can also be considered as an attempt to recover this query generation process.

Note that, besides the input from a user, the word "query" sometimes also indicates an internal representation used by a search system. For example, in the sentence "we use a weighted bag of words as the query for retrieval", the word "query" actually means an internal representation generated by a query reformulation model. In this dissertation, unless otherwise stated, we use the first meaning of "query", i.e., the input from a user.

1.1 Query Reformulation in Information Retrieval

Query reformulation has been studied for decades in information retrieval. Among many reformulation techniques, query expansion is the most successful. Query expansion adds new words and phrases into the original query. These new words and phrases help capture some important concepts missed in the original query and also alternative expressions. Early work of query expansion focused on term association

and automatic thesaurus construction. These approaches used term association measures such as mutual information to find related words from the entire collection. These related words are then organized as a thesaurus used for query expansion. The expanded words generated by these thesaurus-based approaches are only related to a single query word, thus cannot capture the entire query context.

Pseudo relevance feedback methods solved this problem by extracting words and phrases from the top ranked documents in a result list. Several successful pseudo relevance feedback models have been developed. For example, the relevance model approach (Lavrenko and Croft 2001) adds new words to the original query, the sequential dependence model (Metzler and Croft 2005) adds phrase structure, and the latent concept expansion model (Metzler and Croft 2007) adds new term proximity features and words. Generally, query expansion reformulates the original query as a large, possibly weighted, "bag of concepts", where a concept could be the word and phrase from the original query or a new word and phrase.

However, query expansion does not consider how these concepts would fit together to express the information need. In other words, they cannot discriminate between a good combination of concepts and a bad one. For example, our implementation of the relevance model expands the original query "oil industry history" as a weighted bag of words "0.44 industry, 0.28 oil, 0.08 petroleum, 0.08 gas, 0.08 county, 0.04 history, ..."[2], where the score before each word denotes the probability of generating this word. Using this model, the probability assigned to "oil industry history" is smaller than the probability assigned to "oil industry county", since the generation probability of "history" is smaller than "county". Yet, the former is the original query, while the latter is a somehow random combination of words. Thus, a document containing "oil

[2]The details of the relevance model will be described later.

industry history" will incorrectly receive lower score than a document containing "oil industry county".

From a generative view, query expansion independently selects concepts to generate queries. Since the model itself does not impose any constraint between concepts, besides realistic queries, many obviously artificial queries are also generated, sometimes even with high probabilities. Here, a realistic or actual query is a semantically coherent query, which is posed by a user and is likely to be observed in query logs. In contrast, an artificial query is a random combination of several query concepts, which does not convey a clear information need. For example, "oil industry history" is a realistic query, while "oil industry county" is an artificial query. These artificial queries have a negative effect on the retrieval performance.

Thus, the realistic or actual queries impose important query-level dependencies between query concepts. Query expansion fails to capture this type of dependency. Note that these query-level dependencies are different with some local dependencies captured by phrases or proximity features. Since a phrase or a proximity feature is difficult to express the information need alone, we still need to consider how to use them with other concepts.

1.2 Query Reformulation in Web Search

The popularity of web search engines has recently made large scale query logs available. Query logs record users' search behavior and provide a valuable resource for query reformulation.

Query expansion has been revisited in the web search context. Instead of analyzing the target corpus, new words or phrases are now generated from query logs. Specifically, the queries that either have a significant overlap with the original query or receive a similar click pattern as the original one are collected. The new words and

phrases from these queries are not directly used for retrieval in web search. Instead, these queries are displayed to users as suggestions.

Several new reformulation operations have also been studied in web search. *Query segmentation* (Bergsma and Wang 2007) tries to detect underlying concepts in keyword queries and annotate those concepts as phrases. For example, given the query "oil industry history", query segmentation techniques may detect "oil industry" as a concept and annotate it as a phrase in the new query "(oil industry) history". *Query substitution* (Jones *et al.* 2006) tries to change some words of the original query to bridge vocabulary mismatch. For example, the query "oil industry history" could be changed to "petroleum industry history", since some relevant documents may contain "petroleum industry" instead of "oil industry". *Query deletion* (Jones and Fain 2003) tries to delete extraneous words from the original query. This operation is particularly useful when the original query is long and contains noisy information.

In general, the reformulation models in web search transform the original query into a single reformulated query using a specific reformulation operation. Compared with query expansion in information retrieval, these approaches explicitly model the generation of a reformulated query. On the other hand, these models only output the best reformulated query. Thus, they do not capture the information about alternative reformulated queries. Furthermore, these models only study a specific operation and none of this research considers combining different operations from a unified perspective.

1.3 Interactive vs. Automatic

A query reformulation model can be incorporated into the search system either in an interactive way or in an automatic way. Using the interactive approach, the output of a reformulation model, whether it is a set of words or phrases from query expansion or a reformulated query from a web reformulation model, is displayed to

a user. These words, phrases, or queries will not be added into the retrieval model unless a user explicitly clicks on them. In contrast, the automatic approach does not require any user interaction. Sometimes, a user may not even notice the existence of a reformulation process. The reformulation model will be seamlessly incorporated into the retrieval model.

For example, the query suggestion model works interactively, where the related queries are displayed to users. These queries are not used for retrieval unless a user clicks on any of them. On the other hand, the pseudo relevance feedback model works automatically. The original query is first expanded using the new words picked from the top ranked documents and then this expanded query is fed into the retrieval model without interaction with the user.

In this dissertation, we focus on using the reformulation model in an automatic way. Thus, seamlessly combining the reformulation model with the retrieval model is a critical issue.

1.4 Motivations

Based on the analysis of previous studies on query reformulation, the motivations of this dissertation come from three aspects.

First, a realistic or actual query should be the basic unit of a reformulation model. The query expansion model considers a concept as the basic unit. In order to formulate a new query, this model independently selects several concepts. Thus, besides realistic queries, many artificial queries are also generated, which have some negative effect on the retrieval performance. By considering a realistic query as the basic unit, the important query-level dependencies between concepts are explicitly modeled. In this way, the query generation process can be modeled as sampling a query from a set of realistic queries according to some probabilistic distribution. Another reason using a realistic query as the basic unit is to use the important features provided in

query logs. For example, the frequency of a query observed in query logs provides a reasonable measure for its popularity and the number of clicked documents after posing a query is related to its quality. All these features characterize a realistic query as a whole, thus it is difficult for query expansion models to use these features.

Second, different query reformulation models have been developed to address a specific reformulation operation such as query segmentation, query substitution and query deletion. Due to the complexities of a user query, applying a single reformulation operation is usually not enough to achieve satisfactory retrieval performance. Thus, we need to combine different reformulation operations within the same framework. Instead of generating a single reformulated query using a specific operation as most web reformulation models have done, we should apply more operations to the original query and also generate more alternative queries.

Third, most of previous research considers query reformulation as a query processing step which is independent from the following retrieval step. For example, the relevance model (Lavrenko and Croft 2001) generates the expanded query representation based on the "relevance" of each word. Some web reformulation techniques (Bergsma and Wang 2007; Jones *et al.* 2006) train their models according to the judgments of human annotators, who are asked to decide whether a reformulated query is "good" or not according to their knowledge. The relevance of words and the goodness of reformulated queries are both potentially important indicators for retrieval performance, but they may not necessarily lead to good retrieval performance, especially considering that different retrieval models can have widely different performance for a given set of words or queries. Therefore, it is essential to jointly consider the reformulation model and the retrieval model. In other words, the reformulation model should be trained by directly optimizing the performance of the retrieval model that will be used. There has, however, been little prior research in this direction.

1.5 A Query Distribution Model

In this dissertation, we propose a novel reformulation framework where the original query is transformed into a distribution of reformulated queries. A reformulated query is generated by applying different operations including adding or replacing query words, detecting phrase structures, and so on. First, since the reformulated query is a realistic or actual query that involves a particular choice of words and phrases, this framework captures important query-level dependencies. Second, this framework naturally combines query segmentation, query substitution and other possible reformulation operations, where all these operations are considered as methods for generating reformulated queries. In other words, a reformulated query is the output of applying single or multiple reformulation operations. The probabilities of alternative reformulated queries can then be estimated within the same framework. Third, the probabilities assigned to each reformulated query are estimated by directly optimizing the retrieval performance on the training set. In this way, the reformulation model and the retrieval model is jointly considered. The reformulation model can be adapted to the retrieval model that will be used.

This framework potentially provides a better way of modeling users' reformulation behavior. During a search session, the user will pose a series of reformulated queries if the original one doesn't work well, thus modeling reformulation as a distribution of related queries instead of a large bag of related words seems more natural.

The query distribution model consists of three major components. First, a set of reformulated queries are generated, where each query is the output of applying a single or multiple reformulation operations. Second, the probability is estimated for each reformulated query by directly optimizing the retrieval performance. Third, the generated reformulated queries and their associated probabilities are used for retrieval. The retrieval model is a weighted combination of the retrieval scores using each reformulated query and their corresponding probabilities serve as weights.

1.6 Contributions

Our major contributions through this dissertation are as follows:

1. **Novel Reformulation Framework.** We propose a novel framework for query reformulation, where the original query is transformed into a distribution of reformulated queries and the retrieval model is considered as an integrated part of this framework.

2. **Novel Probability Estimation Approaches.** We propose two approaches to estimate the probability of each reformulated query by directly optimizing the retrieval performance. These probability estimation approaches consider the reformulation model and the retrieval model jointly.

3. **Hierarchical Query Distribution for Complex Queries.** We propose a hierarchical query distribution that extends the standard query distribution model to capture the dependencies between the reformulated queries. This model is useful for dealing with complex queries that require a series of reformulation operations.

4. **Effective Reformulated Query Generation Approaches.** We propose several effective approaches to generate the reformulated queries based on the type of the original query.

5. **State of the art retrieval performance.** We test the query distribution model on different tasks such as *ad hoc* retrieval and web search and achieve consistent and significant performance improvement over state-of-the-art techniques.

1.7 Organization

In the rest of this dissertation, we first review previous work on query reformulation, including the work in information retrieval and the work on web search (Chapter

2). Then, we introduce the framework of the query distribution model and compare it with previous reformulation models (Chapter 3). Following that, we describe four approaches of generating reformulated queries according to different types of queries and applications (Chapter 4). We also propose two probability estimation methods that learn the probabilities assigned to the reformulated queries by directly optimizing their retrieval performance (Chapter 5). Two applications are introduced to show the effect of the query distribution model on *ad hoc* retrieval using short and long queries, respectively (Chapter 6 and Chapter 7). The hierarchical query distribution model is proposed to process complex queries and model the relationships between the reformulated queries (Chapter 8). Finally, we conclude this dissertation and discuss some future directions (Chapter 9).

CHAPTER 2

RELATED WORK

Query reformulation has been an important topic in information retrieval for long time. Recently, it has also attracted much attention in web search, since it significantly improves web users' search experiences. However, to the best of our knowledge, there is little work that models reformulation as a distribution of reformulated queries, especially considering the retrieval model as an integrated part of the approach. In this chapter, we will first review previous research in information retrieval and then describe some recent work in web search. A distribution-based representation is also described. Furthermore, we also consider previous work that weights words and phrases by directly optimizing the retrieval performance. Finally, previous techniques that model the relationships between reformulated queries are reviewed.

2.1 Query Reformulation in Information Retrieval

Some standard query processing techniques such as stemming, stopword removal and spelling correction can all be considered as cases of query reformulation, since they modify the original query to improve retrieval performance.

Stemming transforms a word into its root form, which helps a search system to match variants of a query word in documents. The Porter stemmer (Porter 1980) is the most popular algorithmic stemmer. It uses a series of rules to remove the suffixes of a word. Different variants of a word cannot be fully captured by an algorithmic stemmer using a set of rules, thus a dictionary-based stemmer simply store the word variants in a dictionary to avoid the errors of the algorithmic rules.

It is also reasonable to combine both types of stemmers and the Krovetz stemmer (Krovetz 1993) provides a good example of this combination strategy. Instead of stemming the entire collection, the query-based stemming (Xu and Croft 1998) treats stemming as a query expansion process, where the new words are limited to word variants.

Stopword removal is a basic query processing technique. The INQUERY stopword list (Allan *et al.* 2000) is the most common stopword list used in information retrieval. Lo et al (2005) proposed automatically constructing stopwords from the corpus and the general idea is to select words with high *idf* scores. In addition to stopwords, Callan and Croft (1993) removed stop phrases. Recently, Huston and Croft (2010) studied removing stopwords and stop phrases in verbose queries and proposed an approach to automatically detect stop phrases.

Spelling correction is another important query processing step, since many user queries contain spelling errors. String edit distance is typically used to generate candidate corrections and a noisy channel model provides a general framework for spelling correction. More details can be found in (Kukich 1992; Jurafsky *et al.* 2000).

Besides the above query processing techniques, query expansion has been widely studied in information retrieval as a query reformulation technique.

Some early work of query expansion focused on term association and automatic thesaurus construction. These studies analyze the entire collection to group the related words together and organize them into a thesaurus. Van Rijsbergen (1979) described several term association measures. Simply expanding a query word using its related words in a thesaurus does not work well, since these related words cannot capture the query context. Some approaches (Crouch and Yang 1992; Qiu and Frei 1993; Jing and Croft 1994) carefully designed how to construct a thesaurus and select expanded words and reported good retrieval performance. Since

the thesaurus-based approaches analyze the entire collection, they are usually called "global analysis" techniques.

Compared with global analysis techniques that use the entire corpus, researchers have noticed that the top ranked documents in a result list provide a better approximation of the query context. This type of technique referred to as "local analysis". Another name for this technique is pseudo relevance feedback that may be more familiar to readers. This name comes from a series of classic query expansion methods (Rocchio 1971) that require a user to provide feedback for the relevance of the top ranked documents. In local analysis, the top ranked documents are assumed as relevant and no relevance feedback is required. Thus, local analysis is also called pseudo relevance feedback.

Xu and Croft (2000) is among the earliest work that compares local analysis techniques with global analysis techniques. Laverenko and Croft (2001) and Zhai and Lafferty (2001a) revisited pseudo relevance feedback in the language modeling framework (Ponte and Croft 1998; Zhai and Lafferty 2001b). Laverenko and Croft (2001) proposed a framework for estimating the concept of relevance, which can be naturally used for query expansion. Zhai and Lafferty (2001a) assumed that the language models of the top ranked documents are mixtures of the relevance model and the background model and estimated the relevance model using the expectation-maximization algorithm. Metzler and Croft (2007) proposed a latent concept expansion model that adds both words and phrases to the original query. Collins-Thompson and Callan (2007) studied the uncertainty in pseudo relevance feedback and considered the relevance model as a distribution. Cao et al (2008) noticed that many expansion words returned by the pseudo relevance feedback model are actually unrelated to or even harmful for the original query. Thus, they proposed a classifier to help decide which terms are good expansion words. Lee et al (2008) used a cluster-based resampling method to select dominant top documents for pseudo relevance feedback, which im-

proves the quality of the pseudo relevant documents. Xu et al (2009) developed a query dependent pseudo relevance feedback model that uses Wikipedia in different ways according to query types. Lang et al (2010) proposed Hierarchical Markov Random Fields (HMRFs) to improve the latent concept expansion model (Metzler and Croft 2007) by considering the hierarchical structure within documents. Lv and Zhai (2010) developed the positional relevance model to capture the term position and proximity in feedback documents. Bendersky et al (2011) considered a parameterized concept weighting method for pseudo relevance feedback.

Besides query expansion approaches, Metzer and Croft (2005) detected phrases from the original query and proposed the sequential dependence model to combine query words, phrases and term proximity features.

In general, previous reformulation models in information retrieval add different query concepts such as words, phrases and term proximity features, to the original query, but they do not consider how these concepts would fit together to form realistic or actual queries. Thus, important query-level dependencies between concepts were ignored. In the proposed query distribution model, a reformulated query is considered as the basic unit, which explicitly models how query concepts are used in realistic queries.

2.2 Query Reformulation in Web Search

Query reformulation is considered an important technique for web search. With the popularity of search engines, large scale query logs have become a valuable resource for many query processing and reformulation techniques. Several reformulation techniques studied in information retrieval have been revisited in the web search scenario, especially with the help of query logs.

Peng et al (2007) proposed a context-sensitive stemming method for web queries, where a query word is stemmed based on the analysis of its query context using query logs. A context-sensitive document matching is also conducted during retrieval.

Sun et al (2010) used query logs for spelling correction. A phrase based error model was trained using the query-correction pairs extracted from the clickthrough data. Compared with other approaches, this model captures the contextual information between query words.

For query expansion, Billerbeck et al (2003) extracted expansion terms from associated queries. Cui et al (2002) proposed a probabilistic query expansion model using query logs. In addition, personalized query expansion (2007) was also studied.

Compared with query expansion, query suggestion has been widely adopted in web search, which finds and suggests semantically related queries for users. Vlachos et al (2004) and Chien and Immorlica (2005) calculated the similarity of queries based on their temporal behaviors. Cucerzan and Brill (2006) measured query similarity with aggregated session statistics. Cao et al (2008) proposed a context-aware query suggestion method, which considered the preceding queries as context and utilized click-through and session data.

Several types of query reformulation operations have been studied in web search, which modifies the original query for retrieval.

Bergsma and Wang's work (2007) is among the earliest studying query segmentation on the web. They trained a SVM classifier to make a decision for each segmentation position using several types of features. Tan and Peng (2008) proposed an unsupervised query segmentation method using a concept-based language model. The parameters of the language model were estimated by the EM algorithm conducted on a partial corpus specific to the query. Bendersky et al (2009) proposed a two-stage query segmentation method.

Jones et al (2006) first provided a clear definition for query substitution on the web. In their work, a substitution pair is generated from two successive queries that share some common terms. Then, a linear regression classifier is trained to decide the quality of each pair. Wang and Zhai (2008) mined query logs to find potential query term substitution and addition patterns. Their basic idea is that similar words would have similar neighborhood words in query logs. Dang and Croft (2010) re-implemented and tested Wang and Zhai's method for TREC collections. The results showed that this method does not work well for the well-formed TREC queries. They also showed that anchor text can be used as a good substitute for real query logs.

Jones et al (2003) proposed a model to predict which word should be removed from the original query in order to increase the query coverage in the web search.

Guo et al (2008) proposed a CRF-based model for query refinement, which combined several tasks such as spelling correction, stemming and phrase detection. Within their model, different tasks can be solved simultaneously instead of sequentially. Their model mainly focused on morphological changes of the query words such as spelling correction and stemming, and did not consider query substitution.

Researchers (Boldi *et al.* 2009; Jansen *et al.* 2009) also studied query reformulation patterns and built a model to automatically classify query reformulations into categories. For example, Boldi et al (2009) classify reformulations into four categories, which are generalization, specialization, error correction and parallel moving.

Little research has considered different reformulation operations from a unified view and studied their effect on improving retrieval performance. In the proposed query distribution model, a reformulated query is the output of applying a single or multiple operations. Thus, different operations are combined within the same framework. Moreover, we will explicitly explore the effect of reformulation operations on improving retrieval performance.

2.3 Distribution-based Query Representation

Collins-Thompson (2008) studied the uncertainty in pseudo relevance feedback and modeled relevance using the feedback model distribution. Specifically, given an input query and a set of top ranked documents, several feedback models, i.e, the language model returned by a pseudo relevance feedback approach, are generated by resampling the top ranked documents. Then, a Dirichlet distribution is estimated using the generated feedback models.

In this dissertation, we model reformulation as a multinomial distribution over the space of realistic queries, while Collins-Thompson (2008) model relevance as a Dirichlet distribution over the space of language models. As aforementioned, the language model returned by a pseudo relevance feedback is a weighted "bag of words", thus it cannot capture the contextual information of realistic queries. Furthermore, the multinomial distribution used in this dissertation is estimated by directly optimizing the retrieval performance.

2.4 Combing Reformulation Model and Retrieval Model

Previous research optimizes the reformulation model and the retrieval model separately, and few of them have considered these two models jointly.

"Learning to rank" techniques are proposed to optimize the retrieval model. This direction has attracted considerable attention recently. The basic idea of the "learning to rank" approaches is to characterize each query-document pair as a set of features and construct a training set using the queries previously observed and their relevance judgments. A ranking model is learned using machine learning techniques. This learned model is then used to rank documents for unseen queries. Several learning to rank methods have been proposed, such as RankSVM (Herbrich *et al.* 2000), RankBoost (Freund *et al.* 2003) and RankNet (Burges *et al.* 2005). These methods optimize the ranking loss for a pair of documents, thus they are called as "pairwise"

methods. Cao et al (2007) extended "pairwise" methods to "listwise" methods that considered a ranked list of documents as an instance used in the ranking model and designed a neural network based approach, i.e., ListNet. AdaRank (Xu and Li 2007) is another listwise approach that uses the boosting method and has the property of directly optimizing the retrieval performance. Qin et al (2008) considered the relationships between documents and proposed a method to learn the retrieval scores of a list of documents simultaneously.

Many reformulation models are optimized using human annotations (Bergsma and Wang 2007; Jones *et al.* 2006). For example, Bergsma and Wang (2007) trained their segmentation model using the best segmentations provided by people. Also, Jones et al (2006) asked human annotators to decide whether a query substitution is good. As aforementioned, these human annotations do not necessarily result in good retrieval performance.

Some recent query weighting approaches (Bendersky *et al.* 2010; Bendersky *et al.* 2011) learned the weighting of query words by directly optimizing retrieval performance. Also, Sheldon et al (2011) proposed merging the search results of query reformulations using a supervised merging method.

2.5 Modeling the relationships between reformulated queries

Most previous research relies on query logs to model the relationships between reformulated queries . Boldi et al (2008) proposed to build a query-flow graph that models web users' search behaviors. Specifically, the edges between two queries indicated that they were likely to belong to the same search mission. Both the time and the textual information was used for the graph construction. The query-flow graph has demonstrated its promise when applied to tasks such as session detection and query suggestion. Mei et al (2009) presented a general framework to model search sequences that are represented as a nested sequence of search objects. Various search

sequence analysis tasks were modeled within this framework and the features were reused across different tasks.

In this dissertation, we consider reformulation sequences that model a series of reformulation operations without using query logs. Also, the reformulation sequences are directly incorporated into the retrieval model.

CHAPTER 3

FRAMEWORK OF QUERY DISTRIBUTIONS

In this chapter, we first describe the framework of the query distribution model. Then, we compare this model with previous reformulation models. Part of the content of this chapter has been published in our previous work (Xue and Croft 2010).

3.1 Framework

Formally, given the original query q, we first generate a set of reformulated queries $S_Q = \{q_r\}$, where q_r is a reformulated query. q_r is the output of applying single or multiple reformulation operations. Note that the original query q also belongs to S_Q, which can be considered as a special reformulated query without applying any reformulation operation. Then, we assign the probability $P(q_r|q)$ to each reformulated query, which measures how likely q_r will be generated from the underlying information need of q. Since the sum of the probabilities is equal to one, i.e. $\sum_{q_r \in S_Q} P(q_r|q) = 1$, $P(q_r|q)$ is considered as a multinomial distribution over S_Q. The query distribution model is denoted as **QDist**.

For example, given the original query "oil industry history", we first generate a set of reformulated queries "(oil industry)(history), (petroleum industry)(history), (oil and gas industry)(history), (oil)(industrialized)(history)...". Here, a reformulated query is generated by first applying query substitution and then applying query segmentation. For example, the original query first has a substitution to produce "petroleum industry history" and then it is segmented to produce "(petroleum industry)(history)". Then, we estimate the probability for each reformulated query.

Table 3.1. The query distribution representation for "oil industry history"

Query Distribution (QDist)
(0.78 oil industry history),
(0.08 (oil industry)(history)),
(0.05 (petroleum industry)(history)),
(0.05 (oil)(industrialized)(history)),
(0.04 (oil and gas industry)(history))...

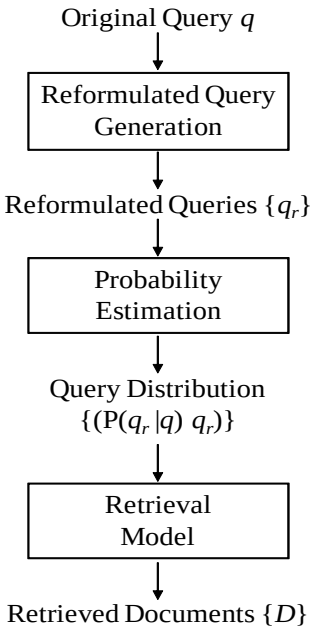


Figure 3.1. The framework of the query distribution model.

The query distribution representation for "oil industry history" is displayed in Table 3.1. This representation consists of a set of tuples and each tuple consists of the reformulated query and the corresponding probability. For example, in the tuple "(0.08 (oil industry)(history))", 0.08 is the probability assigned to "(oil industry)(history))".

Fig. 3.1 shows the framework of the query distribution model, which consists of three major components. The first component generates the reformulated queries $\{q_r\}$ and the second component estimates the probabilities $\{P(q_r|q)\}$. In the third component, the generated query distribution is used by the retrieval model to retrieve documents.

The query generation component generates a set of reformulated queries that convey the same information need as the original query but using different expressions. Different strategies can be adopted according to the type of the original query and the properties of the applications. For example, given a keyword query, using query logs has been shown as an effective way to generate reformulated queries for web search. For other applications where the query log data is not available, analyzing the passages from the target corpus also helps. Beyond keyword queries, natural language questions can be reformulated by searching large scale Q&A archives. In patent retrieval, the whole patent application is transformed into prior-art queries by extracting words and phrases from different fields using different weighting methods.

The probability estimation component assigns the probability to each reformulated query. These probabilities measure the importance of the corresponding queries for retrieval. Some previous models (Bergsma and Wang 2007; Jones *et al.* 2006) estimate these probabilities according to the judgments of human annotators, who are asked to decide whether a reformulated query is "good" or not according to their knowledge. However, the "goodness" of a reformulated query does not necessarily lead to good retrieval performance, especially considering that different retrieval models can have very different performance for a given set of queries. Thus, it is essential to jointly consider the reformulation model and the retrieval model. In other words, the probabilities assigned to reformulated queries should be trained by directly optimizing the performance of the following retrieval model.

The retrieval component takes into the generated query distribution and outputs a set of retrieved documents. This component contains an underlying retrieval model M. Formally, given the query distribution, the retrieval score of a document D is calculated in Eq. 3.1.

$$score(q, D) = \sum_{q_r \in S_Q} P(q_r|q) \log P(q_r|D) \tag{3.1}$$

where $P(q_r|q)$ is the probability of generating q_r from the underlying information need of q and $P(q_r|D)$ is the probability of generating the reformulated query q_r from the document D. $\log P(q_r|D)$ can be considered as the retrieval score of the document D using q_r, which is returned by the underlying retrieval model. Thus, Eq. 3.1 shows that the retrieval score using the query distribution is a weighted combination of the retrieval scores using each reformulated query q_r in S_Q, where the weight is $P(q_r|q)$.

Note that the query distribution model provides a general framework to transform the original query into a set of reformulated queries and assign the probability to each query by directly optimizing the retrieval performance of the underlying retrieval model. Each component can be implemented using different strategies according to applications. In the following chapters, we will describe the approaches of generating reformulated queries and estimating probabilities, respectively.

3.2 Comparison with Previous Reformulation Models

In the rest of this chapter, we compare the query distribution model with two types of reformulation models, which we call the Concept Distribution models and the Single Reformulated Query models, to help readers understand the advantages of the query distribution model.

3.2.1 Concept Distribution (CDist)

The first category of previous reformulation models is Concept Distribution, which is denoted as CDist. Here, a "concept" denotes any component extracted from the original query. Besides the original query words, it also includes query phrases (Metzler and Croft 2005), new words (Lavrenko and Croft 2001; Metzler and Croft 2007), new phrases (Metzler and Croft 2007) and some proximity features (Metzler and Croft 2005; Metzler and Croft 2007). Formally, this category of models is defined as follows.

The Concept Distribution model first generates a set of concepts $S_C = \{c\}$, where c denotes a concept. Then, the probability $P(c|Q)$ is assigned to each concept c, which measures the importance of c. Since the sum of $P(c|Q)$ over all concepts in S_C is equal to one, i.e. $\sum_c P(c|Q) = 1$, $P(c|Q)$ can be considered as a multinomial distribution over S_C. The concept distribution can be represented as a set of tuples $\{(P(c|Q) \;\; c)\}$.

When the concept distribution is used for retrieval, the retrieval score of a document D is calculated in Eq. 3.2.

$$score(q, D) = \sum_{c \in S_C} P(c|q) \log P(c|D) \tag{3.2}$$

where $P(c|q)$ is the probability assigned to the concept c and $\log P(c|D)$ is the probability of generating c from the document D. $\log P(c|D)$ can be considered as the retrieval score of a document D using the concept c as the query. Thus, Eq. 3.2 shows that the retrieval score using CDist is a weighted combination of the retrieval score using each concept c in S_C, where the weight is $P(c|q)$.

Next, we describe two typical models from this category, the relevance model and the sequential dependency model.

The Relevance Model (RM) (Lavrenko and Croft 2001) limits the concept c to the original query words and the new words that are related to the original query. In the relevance model, $P(c|q)$ is estimated by mixing the language models of the top ranked documents. For example, the original query "oil industry history" could be reformulated as "(0.44 industry), (0.28 oil), (0.08 petroleum), (0.08 gas), (0.08 county), (0.04 history), ...", which includes not only words from the original query such as "oil", "industry" and "history", but also new words like "gas" and "petroleum".

The Sequential Dependence Model (SDM) (Metzler and Croft 2005) limits the concept c to the original query words and the bigrams extracted from the original

Table 3.2. Different query representations for the original query "oil industry history"

Model	Output
Concept Distribution (CDist)	
RM	(0.44 industry), (0.28 oil), (0.08 petroleum), (0.08 gas), (0.08 county), (0.04 history), ...
SDM	(0.28 oil), (0.28 industry), (0.28 history), (0.08 (oil industry)), (0.08 (industry history))
Single Reformulated Query (SRQ)	
SEG	(oil industry)(history)
SUB	petroleum industry history
Query Distribution (QDist)	
(0.78 oil industry history), (0.08 (oil industry)(history)), (0.05 (petroleum industry)(history)), (0.05 (oil)(industrialized)(history)), (0.04 (oil and gas industry)(history))...	

query q[1]. In the sequential dependence model, $P(c|q)$ is estimated based on the type of c, i.e., word or bigram. In other words, all words are assigned the same probability value and this is also true for bigrams. For example, the original query "oil industry history" could be reformulated as "(0.28 oil), (0.28 industry), (0.28 history), (0.08 (oil industry)), (0.08 (industry history))". Besides the words from the original query, this also includes bigrams such as "oil industry" and "industry history", which are denoted as "(oil industry)" and "(industry history)".

Table 3.2 shows the representations generated by RM and SDM.

3.2.2 Single Reformulated Query (SRQ)

The second category of previous models is a single reformulated query generated by applying a specific reformulation operation, which is denoted as SRQ.

[1]For each bigram, the sequential dependency model applies two types of window operators, the ordered window and the unordered window. For simplicity, we don't differentiate these two types of bigrams, but the following analysis is still valid if the differentiation is considered.

Formally, $q_r^\star(oper)$ denotes the new query generated after applying the reformulation operation *oper* to the original query q. $q_r^\star(oper)$ is simplified to $q_r^\star$ if the operation *oper* is not explicitly mentioned.

When $q_r^\star$ is used for retrieval, the retrieval score of document D is calculated as follows.

$$score(q, D) = \log P(q_r^\star|D) \tag{3.3}$$

Next, we describe two reformulation operations, query segmentation and query substitution.

Query Segmentation (SEG) (Bergsma and Wang 2007) is the operation of grouping query words into phrases. Given the original query $Q = q_1 q_2 ... q_l$, the segmented query is denoted as $p_1 p_2 ... p_m$. Here, p_i is a phrase, which groups some original query words $q_{j+1} ... q_{j+k}$ together. j indicates the index of the original query and k is the length of the phrase p_i. If k is equal to one, p_i is a single word. For example, "(oil industry)(history)" is a segmentation of the original query.

Query Substitution (SUB) (Jones *et al.* 2006) is the operation of replacing some original query words with new ones. Given the original query $Q = (q_1 ... q_{i+1} ... q_{i+s} ... q_l)$, the substituted query is denoted as $q_1 ... q'_1 ... q'_t ... q_l$, where the original words $q_{i+1} ... q_{i+s}$ are replaced with $q'_1 ... q'_t$. Here, s is the number of the original query words to be replaced and t is the number of new query words. For example, "petroleum industry history" replaces "oil industry" with "petroleum industry". s and t are not necessarily equal so substitution can expand the original query. For example, "oil and gas industry history" substitutes "oil industry" with "oil and gas industry".

Table 3.2 shows the representations generated by SEG and SUB.

3.2.3 Query Generation Process

A query reformulation model can be considered as a process of generating queries. In this subsection, we compare the query generation process of different reformulation models.

The query generation process of the Single Reformulated Query (SRQ) model is deterministic. Given the original query q, it will generate a single reformulated query $q^\star$. This process is shown in Eq. 3.4.

$$P(q_r|q) = \begin{cases} 1 & \text{when } q_r = q_r^\star \\ 0 & \text{when } q_r \neq q_r^\star \end{cases} \tag{3.4}$$

In contrast, the Query Distribution (QDist) model generates queries in a probabilistic way. Specifically, a query q_r is sampled from S_Q according to $P(q_r|q)$, which is a multinomial distribution. Note that both SRQ and QDist generate generate realistic or actual queries, in the sense of being semantically coherent and likely to be found in query logs.

The query generation process of the Concept Distribution (CDist) model is not as straightforward as SRQ and QDist, since the basic unit of this model is a concept. In order to generate a query q_r, CDist will first sample the count of concepts in q_r, i.e. $|q_r|$, according to a count distribution δ and then sample $|q_r|$ concepts from S_C independently according to $P(c|q)$. This process is shown in Eq. 3.5.

$$P(q_r|q) = \delta(|q_r|) \prod_{i=1}^{|q_r|} P(c_i|q) \tag{3.5}$$

Since the generation process of CDist assumes that the concepts are generated independently, this model does not consider how to form realistic queries using concepts. Thus, many artificial queries that will almost never be posed by users are also generated. These artificial queries have negative effect on the quality of the reformulation model. Two examples are provided as follows.

In the first example, given the original query "oil industry history" (AP[2]=5.9) and one artificial query "oil industry county" (AP=2.5), we consider the generation probabilities of these two queries using the RM representation (see Table 3.2), which is a typical example of CDist. The concepts considered in RM are the words in the original query and the expanded words.

$$\begin{aligned} P(\text{oil industry history}) &= \delta(3) \times P(\text{oil}) \times P(\text{industry}) \times P(\text{history}) \\ &= \delta(3) \times 0.28 \times 0.44 \times 0.04 \\ P(\text{oil industry county}) &= \delta(3) \times P(\text{oil}) \times P(\text{industry}) \times P(\text{county}) \\ &= \delta(3) \times 0.28 \times 0.44 \times 0.08 \end{aligned}$$

where $\delta(3)$ denotes the probability of generating a query containing three concepts.

According to RM, the probability assigned to "oil industry county" is even higher than the original query "oil industry history", since RM assumes that the words in a query are independent of each other. However, "oil industry county" is an artificial query and cannot lead to good retrieval performance.

In the second example, two reformulated queries with different effectiveness, namely "(oil industry) history" (AP=4.7) and "(oil industry) industry" (AP=2.0), receive the same generation probability in the SDM model (see Table 3.2). SDM considers the words and the bigrams in the original query as concepts.

$$\begin{aligned} P((\text{oil industry})\ \text{history}) &= \delta(2) \times P((\text{oil industry})) \times P(\text{history}) \\ &= \delta(2) \times 0.08 \times 0.28 \\ P((\text{oil industry})\ \text{industry}) &= \delta(2) \times P((\text{oil industry})) \times P(\text{industry}) \\ &= \delta(2) \times 0.08 \times 0.28 \end{aligned}$$

[2]AP denotes the average precision, which is described in the Appendices.

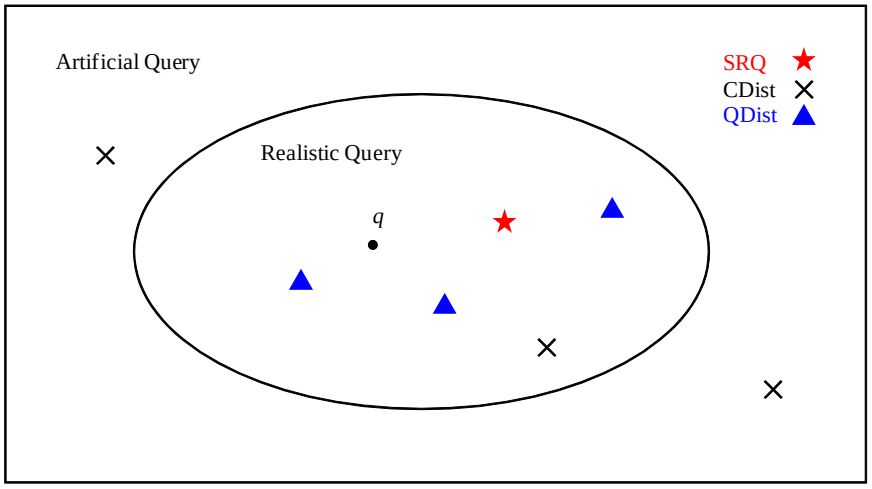


Figure 3.2. The queries generated by different models

where $\delta(2)$ denotes the probability of generating a query containing two concepts.

Without considering whether "history" or "industry" is more likely to be used with the phrase "(oil industry)" to form realistic queries, SDM cannot discriminate between these two queries. On the other hand, "(oil industry) history" is a more realistic query and has better performance than "(oil industry) industry".

Fig. 3.2 illustrates the queries generated by different models. The whole space represents all possible queries generated using different concepts. The space within the circle represents realistic or actual queries posed by users, while the space outside the circle represents artificial queries. The queries generated by CDist spread over the whole space. In contrast, the queries generated by QDist and SRQ will only fall inside the circle.

3.2.4 Retrieval Scores

In this subsection, we further consider the effect of the query generation process on the retrieval scores of different models.

The retrieval score of SRQ (as shown in Eq. 3.3) can be rewritten as follows.

$$
\begin{aligned}
score(q, D) &= \log P(q_r^\star|D) \\
&= \sum_{q_r \neq q_r^\star} 0 \cdot \log P(q_r|D) + 1 \cdot \log P(q_r^\star|D) \qquad (3.6) \\
&= \sum_{q_r} P(q_r|q) \log P(q_r|D) \qquad (3.7)
\end{aligned}
$$

Eq. 3.6 is obtained according to Eq. 3.4, where SRQ assigns all probability to $q_r^\star$. Eq. 3.7 shows that the retrieval score of SRQ can be rewritten as the combination of the retrieval scores using the generated queries.

Before we study the retrieval score of CDist, two claims are made as follows.

Claim 1.

$$
P(c|q) = \sum_{q_r \in \{q_r | c \in q_r\}} P(q_r|q), \textit{ given that } P(q_r|q) = \delta(|q_r|) \prod_{i=1}^{|q_r|} P(c_i|q)
$$

The proof of Claim 1 can be found in the Appendices. Claim 1 shows that using the query generation process of CDist (see Eq. 3.5), the sum of the probability of generating all queries that contain the concept c is equal to the probability of directly generating the concept c.

Claim 2.

$$
P(q_r|D) = \prod_{c \in q_r} P(c|D)
$$

Claim 2 shows that the probability of generating q_r from a document D can be calculated as the product of the probability of generating each concept c in q_r from D. This claim is widely used in the language modeling approach (Ponte and Croft 1998; Zhai and Lafferty 2001b).

Using the above claims, the retrieval score of CDist (as shown in Eq. 3.2) can be rewritten as follows.

$$
\begin{aligned}
score(q, D) &= \sum_{c \in S_C} P(c|q) \log P(c|D) \\
&= \sum_{c \in S_C} \log P(c|D) \sum_{q_r \in \{q_r | c \in q_r\}} P(q_r|q) && (3.8) \\
&= \sum_{q_r} P(q_r|q) \sum_{c \in q_r} \log P(c|D) && (3.9) \\
&= \sum_{q_r} P(q_r|q) \log \prod_{c \in q_r} P(c|D) && (3.10) \\
&= \sum_{q_r} P(q_r|q) \log P(q_r|D) && (3.11)
\end{aligned}
$$

Eq. 3.8 is obtained by using Claim 1. Eq. 3.9 is obtained by changing the order of summation. Eq. 3.11 is obtained using Claim 2. Eq. 3.11 shows that the retrieval score of CDist can be considered as the combination of the retrieval scores using the queries generated by CDist.

In general, Eq. 3.1, Eq. 3.7 and Eq. 3.11 show that the retrieval scores of the three models, i.e. QDist, SRQ and CDist, are closely related to their query generation process. The retrieval score of SRQ is decided by a single reformulated query, while QDist considers a set of reformulated queries. On the other hand, the retrieval score of CDist is affected by both the realistic queries and the artificial queries.

3.3 Summary

In this chapter, we described the framework of the query distribution model and compared it with other reformulation models. Generally, compared with the concept distribution model, the query distribution model considers a realistic query as the basic unit and thus captures the query-level dependencies between concepts. On the other hand, compared with the single reformulated query model, the query distribu-

tion model explores alternative reformulated queries and combines them within the same framework.

CHAPTER 4

REFORMULATED QUERY GENERATION

In this chapter, we focus on the reformulated query generation component. Several approaches are described and each of them is proposed to deal with a specific type of query. First, a passage-analysis technique based on the target corpus is designed for keyword queries. Second, a subset selection strategy is used for verbose queries. Third, we search a Q&A archive to reformulate natural language questions. Fourth, we transform a patent application into prior-art queries. The content of this chapter has been published in our previous work (Xue *et al.* 2010) (Section 4.1), (Xue *et al.* 2008) (Section 4.3) and (Xue and Croft 2009) (Section 4.4)

4.1 Passage Analysis for Keyword Queries

Keyword (or short) queries are the typical type of queries used in information retrieval, where users select some keywords to express their information needs. In web search, large scale query logs have been shown to be successful for reformulating keyword queries. However, these query logs are not available for many important applications such as legal search and forum search. Thus, it is important to develop some techniques to reformulate keyword queries by only using the target corpus and publicly available resources. Furthermore, this technique once available can also be used as a complement to the query log based method.

We propose passage analysis technique based on the target corpus. We observe that in the target corpus, passages containing all query words or most of the query words provide a good source of information for reformulating queries. Specifically,

passages with all query words provide information about the common ways that people split the original query into different concepts. For example, given the original query "oil industry history", the analysis of passages containing all these three words in the Gov2 collection[1] shows that the original query is split as "(oil industry)(history)" in 51 passages. Similarly, passages only containing some query words can indicate possible ways of substituting missing words. For example, the analysis of passages containing "industry history" on Gov2 shows that "petroleum industry history" appears in 46 passages, which indicates that "petroleum industry history" is a potential substitution for the original query.

The passage analysis technique can be divided into two main steps: first, the original query is substituted through passage analysis and with the help of Wikipedia to generate the candidate queries; second, the candidate queries are further segmented based on passage analysis to generate the final reformulated queries.

4.1.1 Generating Query Substitutions

Three different methods are considered to generate query substitutions.

The first method is to use the morphologically similar words. Morphologically similar words are a reliable way to substitute the original query words using appropriate morphological variants. In our work, the morphologically similar words are chosen by considering other query words. Specifically, given the original query $Q = (q_1...q_i...q_l)$, for each query word q_i, we extracted all passages containing all query words except q_i. For each extracted passage, if we find a word q_i' which is morphologically similar to q_i, q_i' will be considered as a substitution of q_i and a candidate query is generated as $Q_c = (q_1...q_i'...q_l)$. Note that a morphologically similar word is considered as a substitution only when it appears in passages containing all other query words, which further guarantees the quality of the substitution word.

[1]Gov2 is a TREC collection used for ad-hoc retrieval.

For example, given the original query "oil industry history", we consider substitutions for each query word. For "history", passages containing both "oil" and "industry" are first extracted. Since some extracted passages contain the word "historical", which is similar to "history", "historical" is used to substitute for "history" and "oil industry historical" is generated as a candidate query. The same process will also be applied to "industry" and "oil", and "industrialized" is found as a substitution for "industry".

A simple method is used to decide whether two words are morphologically similar to each other. Given two words w_i and w_j, the Porter Stemmer (PORTER 1980) is used to get their root forms $stem_i$ and $stem_j$. If $stem_i$ is equal to $stem_j$ or w_i starts with $stem_j$, or w_j starts with $stem_i$, w_i and w_j are considered as morphologically similar, otherwise they are not similar.

The pattern-based method is another way to find query substitutions. Several types of patterns are derived from the original query and these patterns are then used to match qualified passages to find query substitution. Here, two types of patterns are considered, adding-word patterns and changing-word patterns, which will be described in turn.

Adding-word patterns are used to find substitutions which replace a bigram of the original query with an n-gram that adds some words in the middle of the query bigram. Specifically, given the original query $Q = (q_1...q_iq_{i+1}...q_l)$, we consider substitute any q_iq_{i+1} with $q_iw_1...w_sq_{i+1}$. $w_1...w_s$ are the added words and s is the number of added words. Here, we only consider adding one or two words. First, for each bigram q_iq_{i+1}, a pattern is designed as $q_i \star q_{i+1}$, where $\star$ denotes any word or any two words. Then, passages containing all query words $q_1...q_l$ are extracted. For each extracted passage, if the designed pattern $q_i \star q_{i+1}$ matches this passage, q_iwq_{i+1} is collected as a substitution of q_iq_{i+1}. Here, w denotes the added word/words. Thus, a candidate query is generated as $q_1...q_iwq_{i+1}...q_l$.

For example, given the original query "oil industry history", two patterns are derived as "oil $\star$ industry" and "industry $\star$ history". Then, all passages containing all three query words are extracted. For some passages, "oil $\star$ industry" matches expressions "oil and gas industry", thus "oil and gas industry" are considered as a substitution for "oil industry" and a candidate query "oil and gas industry history" is generated.

Changing-word patterns are used to find substitutions that replace a trigram of the original query with a new trigram where the middle word is different. Specifically, given the original query $Q = (q_1...q_iq_{i+1}q_{i+2}...q_l)$, we consider substituting $q_iq_{i+1}q_{i+2}$ with q_iwq_{i+2}, where w is a different word. First, for each trigram $q_iq_{i+1}q_{i+2}$ of the original query, a pattern is designed as $q_i \star q_{i+2}$. Second, passages containing all query words except q_{i+1} are extracted. For each extracted passage, if the pattern $q_i \star q_{i+2}$ matches this passage, q_iwq_{i+2} is collected as a substitution for $q_iq_{i+1}q_{i+2}$ and a candidate query substitution is generated as $q_1...q_iwq_{i+2}...q_l$.

For example, given the original query "oil industry history", a pattern is designed as "oil $\star$ history". Then, all passages containing "oil" and "history" but not "industry" are extracted. For some passages, "oil $\star$ history" matches expressions "oil spill history", thus "oil spill history" is considered as a substitution of "oil industry history".

Some query substitutions are difficult to obtain only relying on corpus information, thus the third method uses Wikipedia as an external resource. A redirect page in Wiki-pedia is designed to send the user to the article with an alternative title[2]. For example, if the user searches for "UK", a redirect page will send the user to the article with title "United Kingdom", since "UK" is the abbreviation of "United Kingdom". Other reasons for maintaining redirect pages include alternative names ("Edi-

[2]The definition of redirect pages and the following examples can be found at `http://en.wikipedia.org/wiki/Redirects_on_wikipedia`

son Arantes do Nascimento" redirects to "Pelé"), less or more specific forms of names ("Hitler" redirects to "Adolf Hitler"), alternative spellings or punctuation ("Colour" redirects to "Color"), likely misspellings ("Massachusets" to "Massachusetts"), plurals ("Gre-enhouse gases" redirects to "Greenhouse gas"), related words ("Symbiont" redirects to "Symbiosis") and so on. Clearly, the redirect page is a valuable resource for query substitution and since they are created by people, the quality is generally good.

All redirect pages are organized as a set of tuples $\{(p_{src}, p_{tar})\}$, where the phrase p_{src} is redirected to the phrase p_{tar}. Given the original query $Q = (q_1...q_{i+1}...q_{i+n}...q_l)$, all n-grams ($n > 1$) are extracted. For each n-gram $q_{i+1}...q_{i+n}$, if it matches p_{src} or p_{tar} in any tuple, the corresponding p_{tar} or p_{src} will be collected as a substitution. Then, a candidate query substitution is generated as $(q_1...p_{tar}...q_n)$ or $(q_1...p_{src}...q_n)$.

For example, for the original query "oil industry history", we extract n-grams "oil industry", "industry history" and "oil industry history". Each of these n-grams is used to match the redirect tuples. The n-gram "oil industry" matches a tuple ("oil industry", "petroleum industry"), thus "petroleum industry" is used as a substitution of "oil industry" and a candidate query "petroleum industry history" is generated.

4.1.2 Generating Query Segmentations

In this step, phrase structures are detected using passage analysis for all candidate queries including both the original query and query substitutions. The basic idea can be described as follows. Given a candidate query, passages containing all query words are extracted. Then, each extracted passage tells us one way to segment the candidate query. After analyzing all extracted passages, the most frequent ways of segmenting the candidate query can be determined.

The details of the algorithm are provided in Fig. 4.1. Generally, the function *DetectSegmention* returns how the input query is segmented in the given passage.

ALGORITHM: Generating Reformulated Query
INPUT: candidate query $Q_c = (q_1 q_2 ... q_m)$, corpus C
OUTPUT: a set of reformulated queries $R = \{(Q_r, Stat)\}$, where Q_r is one way to segment Q_c and $Stat = \{(psg_{id}, doc_{id})\}$ records from which passages (psg_{id}) and documents (doc_{id}), Q_r is detected.
PROCESS:

1. select passages containing $q_1 q_2 ... q_m$ from C.
2. for each selected passage psg
   - get (psg_{id}, doc_{id}), the passage id and corresponding document id of psg.
   - $Q_r = DetectSegments(Q_c, psg)$
   - add $(Q_r, (psg_{id}, doc_{id}))$ into R.

FUNCTION: *DetectSegmention*
INPUT: query $Q_c = (q_1 q_2 ... q_m)$, passage $psg = w_1 w_2 ... w_g$
OUTPUT: Q_r
PROCESS:

1. $S = \emptyset$, $i = 1$
2. while $i \leq g$
   - search the longest string $str = w_i w_{i+1} ... w_{i+s}$ that starts with w_i and matches a substring of $q_1 q_2 ... q_m$.
   - if str is found

     $S \leftarrow str$, $i = i + s + 1$

     else

     $i = i + 1$
3. for each str in S
   - if str is a substring of another string in S

     $S = S - str$
4. According to S, Q_c is segmented to form Q_r.

Figure 4.1. Algorithm for generating reformulated queries.

Table 4.1. Examples of reformulated queries generated using the passage analysis technique

source	controlling type ii diabetes	pyramid scheme
Orig	(controlling)(type ii diabetes)	(pyramid scheme)
Wiki	(controlling)(type 2 diabetes)	(1)(up)(system)
Morph	(control)(type ii diabetes)	n/a
Pat-add	n/a	(pyramid promotional scheme)
Pat-chg	(controlling)(type 2 diabetes)	n/a
source	low white blood cell count	Enron California energy crisis
Orig	(low white blood cell count)	(Enron)(California energy crisis)
Wiki	(low)(leukocyte count)	(Enron)(California electricity crisis)
Morph	(lower)(white blood cell count)	(Enrons)(California energy crisis)
Pat-add	n/a	n/a
Pat-chg	(low red blood cell count)	(Enron)(California electricity crisis)
source	hybrid alternative fuel cars	ban human cloning
Orig	(hybrid)(alternative fuel)(cars)	(ban)(human cloning)
Wiki	(hybrid)(alternative fuel)(vehicle)	(ban)(human)(clone)
Morph	(hybrid)(alternative fueled)(cars)	(bans human cloning)
Pat-add	n/a	(ban on human cloning)
Pat-chg	n/a	(ban reproductive cloning)

Then, the algorithm records all returned segmentations and their associated passage id (psg_{id}) and document id (doc_{id}), which can be used to weight different segmentations.

Some examples are provided to show how the function *DetectSegmentation* works. The input query is "oil and gas industry history", which is a query substitution. Given the passage "...shape the history of the oil and gas industry in Oklahoma during the early days of the Oklahoma Oil boom...", S = {"history", "oil and gas industry", "oil"} after Step 2. Since "oil" is a substring of "oil and gas industry", "oil" is removed from S in Step 3 and S becomes {"history", "oil and gas industry"}. According to S, a segmentation is formed as "(oil and gas industry)(history)".

4.1.3 Examples

Table 4.1 shows examples of the reformulated queries generated using the passage analysis techniques. "Orig" denotes the segmentation of the original query without substitution. "Wiki" denotes the reformulated queries generated from the Wikipedia

redirect page. "Morph" denotes the reformulated queries generated from morphologically similar words. "Pat-add" denotes the method of using adding-word patterns and "Pat-chg" denotes the method of using changing-word patterns. Note that for each substitution method, only one example is displayed, but actually more than one reformulated query would typically be generated.

Table 4.1 contains many interesting reformulated queries. "(controlling)(type 2 diabetes)" reformulates "controlling type ii diabetes", "1-up system" reformulates "pyramid scheme", "(low)(leukocyte count)" reformulates "low white blood cell count", "(Enron)(California electricity crisis)" reformulates "Enron California energy crisis" and so on. Sometimes, the reformulated queries generated from different sources happen to be the same. For example, Wiki and Pat-chg both generate "(controlling)(type 2 diabetes)". Also, some sources cannot generate reformulated queries for certain queries (denoted as "n/a" in Table 4.1).

4.2 Subset Selection for Verbose Queries

The use of verbose (or long) queries helps users to express their information need naturally and saves efforts in choosing keywords. Previous work (Bendersky and Croft 2008; Kumaran and Carvalho 2009), however, has shown that current search engines cannot handle verbose queries well. Thus, dealing with verbose queries poses a new challenge for information retrieval.

In order to extract the important information from the verbose query, we select a subset of query words. In other words, we reformulate the verbose query as a set of subset queries. If the length of the query is n, the total number of subset queries is 2^n, since for each query word, we need to decide whether it should be kept or not.

Considering the number of potential subset queries, some strategies are required to filter the subset queries. Two simple strategies are used according to the study of Kumaran and Carvalho (2009). First, if the length of a query is more than 10 words,

Table 4.2. Examples of the subset queries selected from a verbose query

q: *give information on steps to manage control or protect squirrels*
steps protect squirrels
steps control squirrels
steps control protect squirrels
steps manage squirrels

all query words are first ranked by their *idf* scores and then only the top 10 words are used to generate subset queries. Second, we only keep the subset queries with length between three to six words, since either too long or too short queries usually cannot generate good retrieval performance.

Kumaran and Carvalho (2009) also suggest some other useful filtering strategies, which are currently not used in this dissertation. For example, only the subset queries containing the named entities are kept. Furthermore, all subset queries can be ranked by their quality predictor values and only the top ranked queries are kept.

Table 4.2 shows an example of the subset queries selected from a verbose query. These subset queries keep the important keywords such as "steps", "control", "protect" and "squirrels" and at the same time remove the noisy information such as "give information on".

4.3 Q&A Search for Natural Language Questions

Natural language questions can be considered as special cases of verbose queries. Thus, the subset query selection strategy discussed in previous section can also be applied here. On the other hand, we notice that various natural language questions can be formulated to express the same information need. Thus, it is very interesting to reformulate the original question into a set of alternative questions using different vocabularies and expressions.

In this section, we generate these alternative questions by searching the large scale question and answer archives (Q&A archive). These archives include the FAQ

archives constructed by companies for their products and the archives generated from Web services such as Yahoo Answers! and Live QnA, where people answer questions posed by other people. Since these archives contain a large number of questions that cover a variety of topics, they provide valuable resources to discover alternative questions.

The major challenge for searching the Q&A archive is the word mismatch between the user's question and the questions in the archive. For example, "what is francis scott key best known for?" and "who wrote the star spangle banner?" are two very similar questions, but they have no words in common. This problem is more serious for Q&A retrieval compared with document retrieval, since the question-answer pairs are usually short and there is little chance of finding the same content expressed using different wording.

To solve the word mismatch problem, we use translation-based approaches since the relationships between words can be explicitly modeled through word-to-word translation probabilities. The word-to-word translation probabilities are estimated by considering the question-answer pairs as the "parallel corpus". Furthermore, the corresponding answer part of a question provides complementary information, thus considering the answer part also helps solve the word mismatch problem.

4.3.1 Retrieval Model for Q&A Archives

A typical Q&A archive consists of a large number of question-answer pairs. Here, C denotes the whole archive, $C = \{(Q, A)\}$. (Q, A) denotes a question and answer pair, where Q is a question and A is an answer. Given a natural language question q, the task is to estimate $P((Q, A)|q)$, i.e., the probability of observing (Q, A) given q. The estimation of $P((Q, A)|q)$ is decided by the estimation of $P(q|(Q, A))$. Here, we propose a translation-based language model (TransLM+QL) to calculate $P(q|(Q, A))$ according to the following equations.

$$P((Q,A)|q) \propto P(q|(Q,A)) \tag{4.1}$$

$$P(q|(Q,A)) = \prod_{w \in q} P(w|(Q,A)) \tag{4.2}$$

$$P(w|(Q,A)) = \frac{|(Q,A)|}{|(Q,A)|+\lambda} P_{mx}(w|(Q,A)) + \frac{\lambda}{|(Q,A)|+\lambda} P_{ml}(w|C) \tag{4.3}$$

$$P_{mx}(w|(Q,A)) = \alpha P_{ml}(w|Q) + \beta \sum_{t \in Q} P(w|t) P_{ml}(t|Q) + \gamma P_{ml}(w|A) \tag{4.4}$$

In the above equations, q is the query, w is a query word, (Q, A) is a question and answer pair, C is the background collection, λ is the smoothing parameter, $|(Q, A)|$ is the length of (Q, A).

$P_{ml}(w|C)$ is the background probability of generating w, which is calculated in Eq. 4.5.

$$P_{ml}(w|C) = \frac{\#(w, C)}{|C|} \tag{4.5}$$

where $\#(w, c)$ is the frequency of w in the collection and $|C|$ is the total number of words in the collection.

$P_{ml}(w|Q)$ is the probability of generating w from Q and $P_{ml}(w|A)$ is the probability of generating w from A. They are calculated in Eq. 4.6.

$$P_{ml}(w|D) = \frac{\#(w, D)}{|D|} \tag{4.6}$$

where $\#(w, D)$ is the frequency of w in the document D and $|D|$ is the total number of words in the document. $P_{ml}(w|Q)$ and $P_{ml}(w|A)$ can be calculated by considering the question Q and the answer A as the document D, respectively.

$P(w|t)$ is the word-to-word translation probability. The estimation of $P(w|t)$ is described in the following section.

The most interesting part comes from Eq. 4.4, which consists of three components: $P_{ml}(w|Q)$, $\sum_{t \in Q} P(w|t)P_{ml}(t|Q)$ and $P_{ml}(w|A)$. $P_{ml}(w|Q)$ uses the maximum likelihood estimation. Using this estimation, only the questions containing the exact word

w will generate w with high probabilities. In this way, those questions using different vocabularies to express the same information need will not generate high probability for w. In order to solve this problem, the component $\sum_{t \in Q} P(w|t)P_{ml}(t|Q)$ is introduced, which uses a translation-based estimation. Using this estimation, every word t in the question has some probability of being "translated" into a target word w and these probabilities are added up to calculate the sampling probability. Therefore, if a question has many semantically related words to a target word, then the target word also gets high probability from this question. This sampling approach considers word-to-word relationships and helps to overcome the word mismatch problem. Furthermore, the answer part A provides complementary information to the question part Q. Thus, it is helpful to incorporate the component $P_{ml}(w|A)$, which conducts the maximum likelihood estimation on the answer part.

4.3.2 Learning Word-to-Word Translation Probabilities

The estimation of $P(q|(Q, A))$ heavily depends on the quality of the learned word-to-word translation probabilities $P(w|t)$. Techniques for estimating word-to-word translation probabilities are discussed in this part.

IBM translation model 1 (Brown *et al.* 1993) incorporated an EM-based algorithm to learn the word-to-word translation probabilities. Suppose there is a parallel corpus consisting of English-French sentence pairs, $S = \{(\mathbf{e}_1, \mathbf{f}_1), (\mathbf{e}_2, \mathbf{f}_2), ..., (\mathbf{e}_N, \mathbf{f}_N)\}$. The translation probability from an English word e to an French word f is calculated as:

$$P(f|e) = \lambda_e^{-1} \sum_{i=1}^{N} c(f|e; \mathbf{f}_i, \mathbf{e}_i) \tag{4.7}$$

$$c(f|e; \mathbf{f}_i, \mathbf{e}_i) = \frac{P(f|e)}{P(f|e_1) + ... + P(f|e_l)} \#(f, \mathbf{f}_i) \#(e, \mathbf{e}_i) \tag{4.8}$$

Here, $\lambda_e = \sum_f \sum_{i=1}^{N} c(f|e; \mathbf{f}_i, \mathbf{e}_i)$ is a normalization factor to make the sum of translation probabilities for the word e equal to 1. $\{e_1, ..., e_l\}$ are English words that

appear in $\mathbf{e}_i$. $\#(f, \mathbf{f}_i)$ and $\#(e, \mathbf{e}_i)$ are the number of times the French word f appears in $\mathbf{f}_i$ and the number of times the English word e appears in $\mathbf{e}_i$.

Given the initial value of $P(f|e)$, Eq. 4.7 and Eq. 4.8 are used to calculate the updated $P(f|e)$ repeatedly until the probability converges. Brown et. al. (1993) showed that this process converges to the same final probability no matter what initial values are set.

In a Q&A archive, question-answer pairs can be considered as a type of parallel corpus, which is used for estimating word-to-word translation probabilities. In IBM translation model 1, English is the source language and French is the target language. Since the questions and answers in a Q&A archive are written in the same language, the word-to-word translation probability can be calculated through setting either as the source and the other as the target. $P(A|Q)$ is used to denote the word-to-word translation probability with the question as the source and the answer as the target. $P(Q|A)$ is used to denote the opposite configuration.

For a given word, the related words differ when it appears in the question or in the answer. For example, when the word "cheat" appears in the question part, words such as "trust", "forgive", "dump" and "leave" usually appear in the corresponding answer part. These words represent the answerer's suggestion when the asker poses some question about how to react to cheating behaviors. On the other hand, when the word "cheat" appears in the answer, words such as "husband" and "boyfriend" will be observed in the question, which implies most cheating related questions are about the asker's husband and boyfriend. Clearly, all these words are useful to attack the word mismatch problem, thus it is reasonable to combine $P(Q|A)$ and $P(A|Q)$ instead of choosing just one of them.

In addition, the correspondence of words in the question-answer pair is not as strong as in the English-French sentence pair, thus noise will be inevitably introduced for both $P(Q|A)$ and $P(A|Q)$. Suppose a word w_2 appears in the corresponding

answer or question part whenever the word w_1 appears in the question or answer part. Another word w_3 only appears in the corresponding answer part when w_1 appears in the question part. Intuitively, w_2 should be more similar to w_1 than w_3. This intuition will be considered implicitly by combining $P(Q|A)$ and $P(A|Q)$, since $P(w_2|w_1)$ will get contributions from both $P(Q|A)$ and $P(A|Q)$, but $P(w_3|w_1)$ only gets the contribution from $P(A|Q)$.

Two methods are used to combine $P(Q|A)$ and $P(A|Q)$. These two methods differ in the stage that the combination occurs. The first method linearly combines the trained word-to-word translation probabilities, which is shown as follows:

$$P_{lin}(w_i|w_j) = (1-\delta)P(w_i, Ques|w_j, Answ) + \delta P(w_i, Answ|w_j, Ques) \tag{4.9}$$

The second method first pools the question-answer pairs used for learning the probabilities $P(A|Q)$ and the answer-question pairs used for learning the probabilities $P(Q|A)$ together, and then uses the translation model to learn the combined word-to-word translation probabilities. Suppose we use the collection $\{(Q,A)_1, ..., (Q,A)_n\}$ to learn $P(A|Q)$ and use the collection $\{(A,Q)_1, ..., (A,Q)_n\}$ to learn $P(Q|A)$, then $\{(Q,A)_1, ..., (Q,A)_n, (A,Q)_1, ..., (A,Q)_n\}$ is used here to learn the combination translation probability $P_{pool}(w_i|w_j)$.

Table 4.3 shows some example word-to-word translations learned using three different ways of estimating the translation probabilities. It can be seen that most top target words are semantically related to the source word, regardless of the estimation method.

To clarify the differences between using questions and answers as sources and targets, consider the word "everest". When this word appears in the question part, the words "29,035", "8,850", "feet" and "height" often appears in the corresponding answers, as shown by the $P(A|Q)$ column, since the user often asks about the height of the mountain everest. On the other hand, if this word appears in the answer part,

Table 4.3. Word-to-word translation probability examples. Each column shows the top 10 target terms for a given source term. TTable denotes the type of the word-to-word translation probability table.

Source	**everest**			**xp**		
TTable	$P(A\|Q)$	$P(Q\|A)$	P_{pool}	$P(A\|Q)$	$P(Q\|A)$	P_{pool}
1	everest	mountain	everest	xp	xp	xp
2	29,035	tallest	mountain	drive	window	window
3	ft	everest	tallest	install	computer	install
4	mount	highest	29,035	click	system	drive
5	8,850	mt	highest	system	pc	computer
6	feet	discover	mt	window	version	system
7	measure	hillary	ft	computer	edition	click
8	expedition	edmund	measure	pc	install	pc
9	height	mountin	feet	program	software	program
10	nepal	biggest	mount	microsoft	98	microsoft

the corresponding question part often contains words such as "tallest", "highest", and "mountain" as shown by the $P(Q|A)$ column, because "everest" is used as the answer to the questions such as "what is the highest mountain?". Furthermore, P_{pool} shows that after combining $P(A|Q)$ and $P(Q|A)$, we can obtain both these important words.

It is also interesting to note that for the source term "xp", the rank of "drive" is higher than "window" according to $P(A|Q)$. However, P_{pool} assigns the opposite order for these two words, since "window" is also among the top words according to $P(Q|A)$ but "drive" is not. Intuitively, "window" should be more similar to "xp" than "drive". This intuition supports our assumption that two words are more similar when they are related with different source-target configurations. Also, our proposed combination method indeed boosts such words implicitly.

4.3.3 Examples

Table 4.4 shows some reformulated questions generated. For example, given the original question "who is the prime minister of india", the reformulated questions include "who is current vice prime minister of india" and "who is the army chief of

Table 4.4. Examples of the reformulated questions.

Query:	who is the leader of india
1	who is the prime minister of india
2	who is current vice prime minister of india
3	who is the army chief of india
4	who is the finance minister of india
5	who is the first prime minister of india
Query:	who made the first airplane that could fly
1	what is the oldest aiirline that still fly airplane
2	who was the first one who fly with plane
3	who was the first person to fly a plane
4	who the first one fly to the spase
5	who the first one who fly to sky

india", which are semantically related to the original question but use quite different vocabularies.

4.4 Patent Transformation For Prior-art Queries

Prior-art search in patent retrieval is to find previously published patents on a given topic. It is a common task when the patent examiner needs to decide whether a patent application is novel or when the searcher in the Intellectual Property Division of commercial companies needs to check whether some techniques have been patented. As a legal document for protecting the invention, a patent has complex structures and technical content, which can create significant challenges for the retrieval system. Furthermore, some additional factors can make the problem even worse. In order to extend the coverage of a patent, the writers often intentionally use vague words and expressions in the claim, which increases the difficulty of capturing the real content of a patent. Also, in order to pass the patent examination, writers tend to develop their own terminologies, which can cause serious word mismatch problems. The combination of these factors make prior-art search significantly different with other search tasks, such as web search.

Currently, patent retrieval systems use a typical keyword search approach, such as the retrieval system of USPTO[3]. In this approach, the success of the prior-art search relies on the quality of the keywords posed by the user. However, due to the length of the patent and the professional knowledge required to understand the content, selecting keywords can be a difficult task.

Therefore, we consider a novel scenario for patent search, where the user directly pose the patent application as the query instead of the keywords. This strategy significantly reduces the burden of users. However, how to transform a whole patent into a set of operational queries poses a new challenge for the retrieval system.

Fig. 4.2 shows an example patent. The <TITLE>, <ABST>, <BSUM>, <DRWD>, <DETD> and <CLMS> tags indicate the title, the abstract, the summary, the description of the figures, the main body and the claim fields, respectively. The <CLAS> field provides the category information. The <UREF> is a citation field, which indicates a patent's prior-art references decided by the patent examiner.

4.4.1 Operational Queries Generation

The operational queries considered are Indri queries (Metzler and Croft 2004)[4]. As a well-developed query language, Indri provides abundant operators to satisfy different search requirements. For example, we can use the field operator to indicate the structure information. It is also easy to provide weights for each word and use the phrase operator.

To transform a patent into Indri queries, we need to consider several factors: *Num*, how many query words should be kept, *Field*, where to extract query words; *Weight*, which weighting method is used; *NP*, whether to use noun-phrases as a complement.

[3]United States Patent and Trademark Office, http://www.uspto.gov/.

[4]The details of the Indri query language can be found in the Appendices.

<DOCNO>05593134</DOCNO>
<TITLE>Magnetically assisted piezo-electric valve actuator</TITLE>
<CLAS>
<OCL>251#129.17</OCL>
<XCL>251#331</XCL>
<FSC>251</FSC>
<FSS>129.01; 129.15; 129.17; 331 </FSS>
<ICL>F16K#3102</ICL>
</CLAS>
<UREF>
<PNO>2596409</PNO>
</UREF>
<ABST>
A piezo-electrically actuated fluid control valve has a nozzle type seat, a magnet opposite from the seat, and a piezoelectric actuator in sheet form between the seat and magnet.
</ABST>
<BSUM>.....</BSUM>
<DRWD>.....</DRWD>
<DETD>..... </DETD>
<CLMS>..... </CLMS>

Figure 4.2. An example patent.

A general transforming algorithm is provided in Fig. 4.3, where the discussed factors are used as parameters.

For *Num*, we consider words from 10-100. For *Field*, we consider six fields of a patent with explicit tags, the title field (ttl), the abstract field (abst), the brief summary field (bsum), the description of the figures (drwd), the detailed text description field (detd) and the claim field (clms). Besides them, we also extract the primary claim field (pclms), which is the most important claim in the claim field. Also, we

ALGORITHM: Transforming Patent to Query
INPUT: *Patent*, *Num*, *Field*, *Weight*, *NP*
OUTPUT: *Query*
PROCESS: Rank words in *Field* according to their *tfidf* scores and then select *Num* top ranked words as the query words. Assign *Weight* to each query word to get $Query_w$. Repeat the above steps for noun-phrases to get $Query_{np}$. If *NP* is true, set *Query* as the combination of $Query_w$ and $Query_{np}$; otherwise set *Query* as $Query_w$.

Figure 4.3. General algorithm for transforming the query patent to an effective search query.

Table 4.5. Possible values of the parameters

Name	Parameter Values
Num	10-100
Field	ttl, abst, bsum, drwd, detd, clms, pclms, all
Weight	*tf*, *tfidf*, *bool*
NP	true, false

Table 4.6. Examples of Search Queries

	Field	*Num*	*Weight*	*NP*
Param Config 1	ttl	3	*bool*	false
Query 1	#weight(1.0 piezo 1.0 assist 1.0 magnetic)			
Param Config 2	ttl	3	*tfidf*	false
Query 2	#weight(3.8 piezo 1.7 assist 1.5 magnetic)			
Param Config 3	abst	3	*tf*	true
Query 3	#weight(0.8 #weight(1.0 piezo 2.0 magnet 3.0 seat) 0.2 #weight(1.0 #1(piezoelectric actuator) 1.0 #1(fluid control valve) 1.0 #1(nozzle type seat)))			

consider the case that the query words or noun-phrases are extracted from the whole patent (all), which ignores the structure information. For *Weight*, we consider using equal weights (*bool*), term frequency (*tf*) and the combination of term frequency and the inverted document frequency (*tfidf*). For *NP*, we consider using noun-phrase (true) or not (false). Table 4.5 shows the possible values of different parameters.

4.4.2 Examples

Table 4.6 shows the Indri queries generated from the example patent (as shown in Fig. 4.2) using different parameter configurations. Query 1 extracts the words from the title field and weights them equally, while Query 2 weights these words using their term frequencies. Query 3 combines the words and the noun phrases extracted from the abstract field.

4.5 Summary

In this chapter, we described four techniques of generating reformulated queries. These techniques are designed for different types of queries such as a short keyword query, a verbose query, a natural language question and a patent query. Note that, as an early attempt of using the whole patent as a query, we currently generate a set of Indri queries as a simulation of the realistic user queries. How to directly generate the realistic queries from a query patent will be studied in the future.

CHAPTER 5

PROBABILITY ESTIMATION

In this chapter, we study how to estimate the probability for each reformulated query. We first introduce the probability estimation framework. Then, two approaches are described in detail, where the first one optimizes the performance of using a single reformulated query, while the second one optimizes the performance of using a set of reformulated queries. Part of the content of this chapter has been published in our previous work (Xue *et al.* 2010) (Section 5.2) and (Xue and Croft 2011) (Section 5.3).

5.1 Framework

We first assume that the probability assigned to each reformulated query is a combination of their query feature values as shown in Eq. 5.1.

$$P(q_r|q) = \frac{\exp \sum_k \lambda_k f_k(q_r)}{Z(q)} \tag{5.1}$$

$$Z(q) = \sum_{q'_r} \exp \sum_k \lambda_k f_k(q'_r) \tag{5.2}$$

where $f_k(q_r)$ denotes the query feature value extracted to characterize a reformulated query q_r and λ_k is the parameter corresponding to f_k. $\theta = \{\lambda_k\}$ denotes all the parameters of the model. $Z(q)$ serves as a normalizer.

Fig. 5.1 shows the framework of probability estimation. In the training phase, given a training query q, a set of reformulated queries $\{q_r\}$ and the relevance judgments of q, i.e., $rel(q)$, are provided. We first extract the query features $\{f_k(q_r)\}$ for

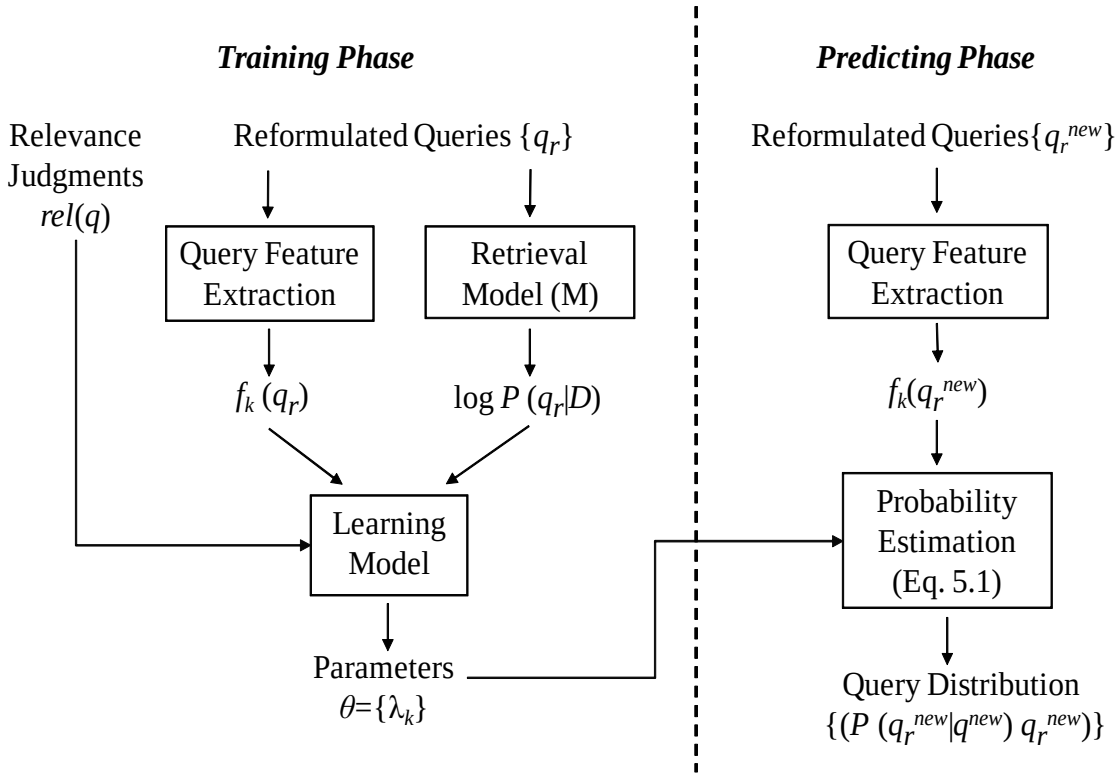


Figure 5.1. The framework for probability estimation

each reformulated query q_r. Then, we run q_r over the collection using the underlying retrieval model M to obtain the retrieval scores of documents $\{\log P(q_r|D)\}$. Taking $\{f_k(q_r)\}$, $\{\log P(q_r|D)\}$ and $rel(q)$ as the input, the learning process will learn the parameters $\{\lambda_k\}$ by directly optimizing the retrieval performance.

In the predicting phase, given an unseen query q^{new} and its reformulated queries $\{q_r^{new}\}$, we first extract the query features $\{f_k(q_r^{new})\}$ for each reformulated query q_r^{new}. Then, we use the learned parameters $\{\lambda_k\}$ to calculate the probability $P(q_r^{new}|q^{new})$ according to Eq. 5.1.

In the rest of this chapter, we describe two learning models. Both of them are designed to directly optimize the retrieval performance of using the reformulated queries.

5.2 Optimizing the performance of using a single reformulated query

Intuitively, we want to learn a model that assigns high probability to the reformulated query with good performance. Since the output of this model is a probability, it can be solved as a standard regression problem, where each reformulated query is treated as an instance and its retrieval performance is treated as the target value. A standard regression method will minimize the distance between the estimated probability of a reformulated query and its retrieval performance as shown in Eq. 5.3.

$$\arg\min_{\theta} \sum_{q} \sum_{q_r} |P(q_r|q) - m(q_r)| \tag{5.3}$$

where $m(q_r)$ denotes the performance measure of q_r, which is calculated using $rel(q)$ and $\{\log P(q_r|D)\}$. Specifically, we first use $\{\log P(q_r|D)\}$ to rank documents and then calculate any standard performance measure such as average precision and precision at the position k using $rel(q)$. $\theta = \{\lambda_k\}$ are the parameters of the model.

However, Eq. 5.3 has two drawbacks. First, Eq. 5.3 does not directly optimize the retrieval performance of training queries. Thus, the learned parameters may not necessarily bring good retrieval performance. Second, the reformulated queries $\{q_r\}$ generated from different training queries q are treated equally. Since some training queries may generate many more reformulated queries than others, the calculation of Eq. 5.3 will be easily dominated by those training queries.

Therefore, we propose a new objective function in Eq. 5.4.

$$\begin{aligned} &\arg\min_{\theta} \quad \text{GM-perf}(\theta) \\ &\text{where} \quad \text{GM-perf}(\theta) = (\prod_{q} \sum_{q_r} P(q_r|q) m(q_r))^{\frac{1}{T}} \end{aligned} \tag{5.4}$$

In Eq. 5.4, T is the number of queries in the training set. $\sum_{q_r} P(q_r|q)m(q_r)$ is the expected retrieval performance using a reformulated query q_r given the original query q. Eq. 5.4 directly optimizes the Geometric Mean of the expected retrieval performance of each query q in the training set. For example, if $m(q_r)$ measures average precision (AP), Eq. 5.4 optimize the geometric mean average precision (GMAP) on the training set, which is a widely used performance measure in information retrieval. According to Robertson (2005), GMAP is sensitive to improvements on the difficult queries.

Since Eq. 5.4 directly optimizes the geometric mean of the retrieval performance of training queries, it solves the first drawback of Eq. 5.3. In addition, for each query in the training set, we calculate the expected retrieval performance over its reformulated queries, which solves the second drawback of Eq. 5.3.

Since T is a constant value after the training set is provided, optimizing Eq. 5.4 is equivalent to optimizing Eq. 5.5.

$$\text{GM-perf}'(\theta) = \prod_{q} \sum_{q_r} P(q_r|q) m(q_r) \tag{5.5}$$

The corresponding log-likelihood expression of Eq. 5.5 is shown in Eq. 5.6.

$$\begin{aligned} l(\theta) &= \sum_{q} \log \sum_{q_r} \exp(\sum_{k} \lambda_k f_k(q_r)) m(q_r) \\ &\quad - \sum_{q} \log Z(q) - \sum_{k} \frac{\lambda_k^2}{2\delta^2} \end{aligned} \tag{5.6}$$

From this, we can show that the partial derivatives of λ_k can be calculated by Eq. 5.7.

$$\begin{aligned} \frac{\partial l(\theta)}{\partial \lambda_k} &= \sum_{q} \sum_{q_r} P_m(q_r|q) f_k(q_r) \\ &\quad - \sum_{q} \sum_{q_r} P(q_r|q) f_k(q_r) - \sum_{k} \frac{\lambda_k}{\delta^2} \end{aligned} \tag{5.7}$$

$P_m(q_r|q)$ is a distribution weighted by $m(q_r)$, which is shown as follows.

$$P_m(q_r|q) = \frac{\exp(\sum_k \lambda_k f_k(q_r)) m(q_r)}{Z_m(q)} \tag{5.8}$$

$$Z_m(q) = \sum_{q_r} \exp(\sum_{k} \lambda_k f_k(q_r)) m(q_r) \tag{5.9}$$

In Eq. 5.7, the first term is the expected value of f_k weighted by the retrieval performance and the second term is the expected value of f_k without weighting. Setting Eq. 5.7 to zero means making the distribution of the reformulated queries correlates the distribution of their retrieval performance.

According to Eq. 5.7, L-BFGS is used for optimization, which is a limited-memory version of the BFGS (Byrd *et al.* 1994) method.

Fig. 5.2 shows the process of optimizing the performance of using a single reformulated query.

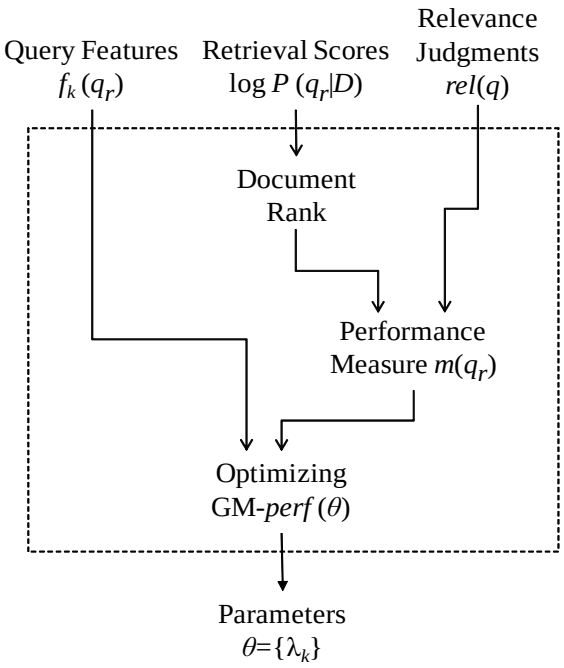


Figure 5.2. The process of optimizing the performance of a single reformulated query.

5.3 Optimizing the performance of using a set of reformulated queries

In the previous section, we optimize the retrieval performance of using a single reformulated query q_r. However, Eq. 3.1 shows that the query distribution model will use a set of reformulated queries $\{q_r\}$ instead of a single one. Thus, there is still some gap between the objective function used in Eq. 5.4 and the retrieval model using the query distribution. In this section, we study how to jointly optimize performance when using a set of reformulated queries.

The general idea of jointly optimizing a set of reformulated queries is to insert their probabilities (Eq. 5.1) into the retrieval model of the query distribution (Eq. 3.1). Then, we directly optimize the retrieval model of the query distribution to obtain the parameters used to calculate the probabilities of the reformulated queries.

In order to facilitate the following inference, we simulate $P(q_r|q)$ using a weight $w(q_r)$ which is calculated in Eq. 5.10.

$$w(q_r) = \sum_k \lambda_k f_k(q_r) \tag{5.10}$$

Compared with Eq. 5.1, Eq. 5.10 removes the normalizer $Z(q) = \sum_{q'_r} w(q'_r)$, which guarantees the definition of the probability. Note that we only use $w(q_r)$ as a substitution of $P(q_r|q)$ for optimization. After we obtain the parameters, we still use Eq. 5.1 to calculate the probability assigned to each reformulated query. Similar assumptions have been made in previous work (Bendersky *et al.* 2010; Wang *et al.* 2010).

We replace $P(q_r|q)$ with $w(q_r)$ in Eq. 3.1 and obtain the following equations.

$$\begin{aligned}
score(q, D) &= \sum_{q_r} P(q_r|q) \log P(q_r|D) \\
&= \sum_{q_r} w(q_r) \log P(q_r|D) \\
&= \sum_{q_r} \sum_k \lambda_k f_k(q_r) \log P(q_r|D) \\
&= \sum_k \lambda_k \sum_{q_r} f_k(q_r) \log P(q_r|D) \qquad (5.11) \\
&= \sum_k \lambda_k F_k(\{q_r\}, D) \qquad (5.12)
\end{aligned}$$

In Eq. 5.12, the retrieval score using the query distribution $score(q, D)$ is rewritten as a linear combination of a set of retrieval features F_k, where F_k and f_k share the same parameter λ_k. Comparing Eq. 5.11 and Eq. 5.12, F_k is calculated as follows.

$$F_k(\{q_r\}, D) = \sum_{q_r} f_k(q_r) \log P(q_r|D) \tag{5.13}$$

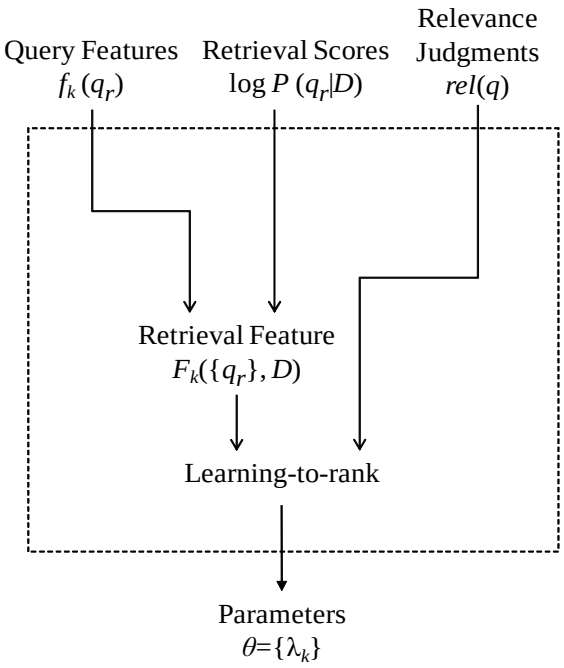


Figure 5.3. The process of optimizing the performance of a set of reformulated queries

Eq. 5.13 shows how to transform f_k into F_k. Basically, F_k is the weighted combination of the retrieval scores using all reformulated queries in $\{q_r\}$, where the weight is the query feature value $f_k(q_r)$.

Next, we need to optimize the performance of the retrieval model as indicated in Eq. 5.12. Many learning-to-rank models have been proposed to directly optimize this type of retrieval model, which is a linear combination of a set of retrieval features. Here, a variant of ListNet (Cao *et al.* 2007) is selected to learn the parameters. Instead of using the neural network, a limited-memory version of BFGS (Byrd *et al.* 1994) is used for optimization due to its efficiency.

Fig. 5.3 shows the process of optimizing the performance of a set of reformulated queries.

5.4 Summary

In this chapter, we described two approaches for probability estimation. Both of them directly optimize the retrieval performance, where the first approach uses a single reformulated query and the second one uses a set of reformulated queries.

CHAPTER 6

APPLICATION I: QUERY DISTRIBUTION FOR SHORT QUERIES

In *ad hoc* retrieval, a user poses a short query, i.e., a small number of keywords, to a search system and the system will return a ranked list of relevant documents. Depending on the application, a document could be a news report or a web page.

Many effective techniques have been developed to improve *ad hoc* retrieval. For example, the query likelihood language model (Ponte and Croft 1998; Zhai and Lafferty 2001b) performs consistently well on a variety of collections. The relevance model (Lavrenko and Croft 2001) is a typical pseudo relevance feedback approach that expands the original query using new words. The sequential dependence model (Metzler and Croft 2005) provides a principled way to incorporate phrases and proximity features. Passage-level evidence has been shown to be useful for *ad hoc* retrieval especially when a document is long and contains multiple topics (Liu and Croft 2002).

6.1 Implementation of Query Distribution Model

In order to implement the Query Distribution Model for short queries, we need to instantiate three components, namely query generation, probability estimation and retrieval model. For query generation, we use the passage analysis technique described in Section 4.1. For probability estimation, we optimize the performance of a set of reformulated queries as shown in Section 5.3. The query features used will be described in this section. For the retrieval model, the query likelihood language

Table 6.1. Implementation of the Query Distribution Model for short queries.

Query Generation	passage analysis (Section 4.1)
Probability Estimation	optimizing a set of queries (Section 5.3)
Retrieval Model	query likelihood language model

model is used. Some extension is also described in this section. The above choices are summarized in Table 6.1.

6.1.1 Query Features

Three types of query features are used to characterize a reformulated query.

The first type of features (PSG) is information extracted from the target corpus. Specifically, we consider the number of passages that contain the whole reformulated query as a feature, which provides evidence about whether this reformulated query is widely used in the target corpus. Different passage sizes are considered. A smaller passage size indicates stronger dependencies between query terms, but has lower coverage since a lot of reformulated queries can not be observed within a tighter window. On the other hand, using a bigger passage size increases the coverage at the cost of sacrificing some quality. Thus, it is interesting to consider features extracted based on different passage sizes. We also consider the number of documents where a reformulated query appears.

The second type of features (NGRAM) is based on query logs and the web corpus. Query logs record the frequencies of the queries used by search engine users, which can be directly used as features of a reformulated query. On the other hand, the web corpus used by search engine companies provides better coverage than the target corpus used for retrieval, and thus can be used as a complement. Furthermore, different fields of a web page serve different purposes, thus additional information can be obtained by splitting the frequencies of a reformulated query in the web corpus according to different fields. Using the Web N-gram Services provided by Microsoft (Huang *et al.* 2010), the above information can be efficiently obtained, where raw

Table 6.2. Three types of features for the reformulated query q_r

PSG Features	
psg$N$-count	count of passages with size N containing q_r
doc-count	count of documents containing q_r
NGRAM Features	
qlog	probability estimated from query logs
title	probability estimated from title
body	probability estimated from body
anchor	probability estimated from anchor text
OPER Features	
Orig	whether it is the original query
Sub	whether it is a substituted query
Morph	whether it is a substituted query using Morph
Pat-add	whether it is a substituted query using Pat-add
Pat-chg	whether it is a substituted query using Pat-chg
Wiki	whether it is a substituted query using Wiki
Seg	whether it is a segmented query

frequencies are simulated by the N-gram language model probabilities. Particularly, these probabilities calculated from query logs and different fields of a web page (body, title and anchor) are provided, respectively.

The third type of features (OPER) indicates the operations applied to the original query to generate the concerned reformulated query. These features correspond to questions such as whether the reformulated query is the original query, or a substituted query, or a segmented query. For a substituted query, we further consider what kind of methods are used (Morphologically Similar Words, Adding Word Patterns, Changing Word Patterns and Wikipedia Redirect Page, see Section 4.1 for details).

The above three types of features are summarized in Table 6.2. psg$N$-count can be instantiated to a variety of features by taking different values of the passage size N.

6.1.2 Retrieval Model

In this subsection, we first describe two types of retrieval models and then discuss how to combine different retrieval models within the query distribution framework.

The query likelihood language model is considered as a document-level model (doc), since it generates query words from the whole document. The details of the query likelihood language model can be found in the Appendices. The retrieval score is calculated using Eq. 6.1.

$$\log P(q_r|D) = \sum_{w \in q_r} P(w|D) \tag{6.1}$$

where w is a query word and $P(w|D)$ is estimated using the language modeling approach (Ponte and Croft 1998; Zhai and Lafferty 2001b). Sometimes, q_r contains a phrase such as "(petroleum industry)(history)". In this case, we treat the phrase "petroleum industry" as a special word.

A passage-level model (psg) is also considered, which prefers documents where the whole query is observed within a passage. The retrieval score is calculated using Eq. 6.2.

$$\log P(q_r|D) = \frac{\#psgN(q_r, D)}{\#psgN(D)} \tag{6.2}$$

where $\#psgN(q_r, D)$ denotes the number of passages with size N containing q_r in document D and $\#psgN(D)$ denotes the total number of passages with size N in document D. This maximal likelihood estimation (Eq. 6.2) can be smoothed with the background model. Based on the value of N, different passage-level models can be generated.

The Indri query language (Metzler and Croft 2004) provides an implementation of the above retrieval models. For example, the Indri queries for a reformulated query "(petroleum industry)(history)" are displayed in Table 6.3 by using the document-level model and the passage-level model, respectively. In this query language, the operator "#combine" is an implementation of Eq. 6.1 and the operator "#uw$N$" is an implementation of Eq. 6.2 where N is the passage size. "#1" is an

Table 6.3. The Indri queries for "(petroleum industry)(history)"

document-level: #combine(#1(petroleum industry) history)
passage-level: #uw20(#1(petroleum industry) history)

operator for a phrase. After running these Indri queries, the retrieval scores returned by the system are $\log(P(q_r|D))$.

Given a set of reformulated queries $\{q_r\}$, different query distributions are generated according to the retrieval model M, since the parameters $\{\lambda_k\}$ learned are dependent on M. Table 6.4 shows the query distributions learned using the document-level model **QDist(doc)** and the passage-level model **QDist(psg)**, respectively. The passage size used in this example is 20. Note that in order to explicitly indicate the retrieval model, we use the Indri queries to represent the reformulated queries in the query distribution.

The document-level model potentially retrieves more relevant documents, while the passage-level model retrieves higher quality documents by imposing more restrictions. Considering their different properties, it is reasonable to combine these two types of models. Suppose that there are p retrieval models. One possible solution is to first generate p query distributions respectively and then linearly combine the generated query distributions. However, it is not easy to decide the combination weights. Instead, an alternative approach is considered in our work. Each query feature f_k is now transformed into p retrieval features $F_k^1 \sim F_k^p$ using different retrieval models. Then, p sets of parameters $\{\lambda_k\}^1 \sim \{\lambda_k\}^p$ are learned. Using different sets of parameters, a reformulated query q_r is expanded as p Indri queries with different probabilities, where each Indri query corresponds to a retrieval model. Since the parameters for different retrieval models are learned within the same framework, the learning procedure automatically takes care of how to balance the weights as-

Table 6.4. Query distributions learned for the original query "oil industry history"

QDist(doc)
0.78 #combine(oil industry history), 0.08 #combine(#1(oil industry) history), 0.05 #combine(#1(petroleum industry) history), 0.05 #combine(oil industrialized history), 0.04 #combine(#1(oil and gas industry) history),...
QDist(psg)
0.59 #uw20(oil industry history), 0.19 #uw20(#1(oil industry) history), 0.12 #uw20(#1(petroleum industry) history), 0.05 #uw20(oil industrialized history), 0.05 #uw20(#1(oil and gas industry) history),...
QDist(doc+psg)
0.804 #combine(oil industry history), 0.054 #combine(#1(oil industry) history), 0.043 #combine(oil industrialized history), 0.042 #combine(#1(petroleum industry) history), 0.032 #combine(#1(oil and gas industry) history), 0.015 #uw20(oil industry history), 0.005 #uw20(#1(oil industry) history), 0.003 #uw20(#1(petroleum industry) history), 0.001 #uw20(oil industrialized history), 0.001 #uw20(#1(oil and gas industry) history),...

signed to different models. Table 6.4 shows the query distribution that combines the document-level model and the passage-level model **QDist(doc+psg)**.

6.2 Experimental Configuration

Three TREC collections, Gov2, ClueWeb (Category B) and Robust04, are used for experiments. The details of these collections can be found in the Appendices. Gov2 and ClueWeb are large web collections with large and varied vocabulary, while Robust04 is a newswire collection that uses a more homogeneous vocabulary. For each collection, two indexes are built, one not stemmed and the other stemmed using the Porter Stemmer (Porter 1980). Stemming transforms a word into its root form, which is conducted either during indexing or during query processing. The latter case treats stemming as a part of query reformulation, which has been shown to be effective for web search (Peng *et al.* 2007). Both cases are considered in the experiments using two types of indexes. No stopword removal is done during indexing. For each topic, the title part is used as the query.

The query set is split into a training set and a test set. On the training set, the parameters λ_k are learned according to Section 5.3. On the test set, the learned parameters λ_k are used to generate the query distribution for each test query and the performance of using the generated query distribution is reported. Specifically, ten-fold cross validation is used, where the query set is randomly split into ten folds. Each time nine folds are used for training and one fold is used for test. This process repeats ten times.

Two passage sizes are used in our work, i.e., 20 and 100, which represent a tight window and a loose window, respectively. Passages are first used to generate reformulated queries as described in Section 4.1. The reformulated queries generated from different passage sizes are put together to form the final set of reformulated queries after removing duplicates. Then, two query features related to the passages are ex-

tracted, i.e. psg20-count and psg100-count (see Table 6.2). For the retrieval models, two passage-level retrieval models are considered, i.e. "#uw20" and "#uw100" (see Table 6.3).

Two types of query distributions are considered: the first one only uses the document-level model ("#combine", see Table 6.3), which is denoted as **QDist(doc)**; the second one combines both the document-level model ("#combine") and two passage-level models ("#uw20" and "#uw100"), which is denoted as **QDist(doc+psg)**.

Several baselines are compared. QL denotes the query likelihood language model (Ponte and Croft 1998; Zhai and Lafferty 2001b). SDM denotes the sequential dependence model (Metzler and Croft 2005). The details of SDM can be found in the Appendices. The parameters of SDM are set according to (Metzler and Croft 2005). RM denotes the relevance model (Lavrenko and Croft 2001). The parameters of RM are set according to (Xu *et al.* 2009). Seg-SVM denotes a SVM-based query segmentation method (Bendersky *et al.* 2009), which is trained on a corpus of 500 pre-segmented noun phrases (Bergsma and Wang 2007). The parameters of Seg-SVM are set according to (Bendersky *et al.* 2009). These methods represent the state-of-the-art reformulation models on TREC collections. Reformulation models used in web search (Jones *et al.* 2006; Wang and Zhai 2008) are not compared here, since they require large scale query logs. QL-psg denotes a passage-augmented language model (Liu and Croft 2002), which augments the retrieval score of a document using its best passage. The same passage sizes (20 and 100) as used in the query distribution model are considered in QL-psg respectively and better performance is reported. The baselines used are summarized in Table 6.5.

The standard performance measures, mean average precision (MAP) and precision at 10 (P10) are used to measure the retrieval performance. The details of the performance measures can be found in the Appendices. In order to improve readabil-

Table 6.5. Summary of Baselines

Name	Description
QL	query likelihood language model
SDM	sequential dependency model
RM	relevance model
Seg-SVM	SVM-based query segmentation
QL-psg	passage-augmented language model

ity, we report 100 times the actual values of these measures. The two-tailed t-test is conducted to measure significance.

6.3 Examples

First, we present examples of the query distribution learned by using the document-level retrieval model QDist(doc). Table 6.6 shows an example on the non-stemmed index. The retrieval performance of using the original query and using the query distribution is compared. The retrieval performance of the top ranked reformulated queries in the query distribution is also displayed.

Table 6.6 shows that the learned query distribution obviously outperforms the original query. In the query distribution, the appropriate probability is assigned to the original query. In most cases, the original query receives the highest probability, since it is not safe to deviate from the original query too much. In some cases, the reformulated queries receive higher probabilities than the original one. For example, given the original query "prostate cancer treatments", the reformulated queries "prostate cancer treatment" and "#1(prostate cancer) treatment" receive higher probability and they both outperform the original query, since "treatment" is more likely to be used with "prostate cancer" in actual queries. "kudzu pueraria lobata" is another example, where the reformulated query "kudzu" receives higher probability, since "pueraria lobata" is another and less popular name of "kudzu" and not very useful for retrieval.

Table 6.6. Example of the query distribution learned on the non-stemmed index using the document-level retrieval model. Top ranked reformulated queries (q_r) are displayed. In the query distribution, the original query (q) is italicized. Average Precision (AP) is reported as the retrieval performance.

$P(q_r\|q)$	Reformulated Query (q_r)	AP
q: prostate cancer treatments		41.21
$QDist(doc)$		50.23
0.1390	#combine(prostate cancer treatment)	42.51
0.1118	#combine(#1(prostate cancer) treatment)	48.88
0.0920	#combine(*prostate cancer treatments*)	41.21
0.0485	#combine(#1(prostate cancer treatment))	11.50
q: cruise ship damage sea life		6.75
$QDist(doc)$		25.83
0.1820	#combine(*cruise ship damage sea life*)	6.75
0.0950	#combine(#1(cruise ship) damage sea life)	21.69
0.0544	#combine(cruise ship damage #1(sea life))	9.80
0.0544	#combine(#1(cruise ship) damage #1(sea life))	7.43
q: kyrgyzstan united states relations		25.88
$QDist(doc)$		39.18
0.1652	#combine(*kyrgyzstan united states relations*)	25.88
0.1128	#combine(kyrgyzstan #1(united states) relations)	36.56
0.0573	#combine(kyrgyzstan united states foreign relations)	14.99
0.0566	#combine(kyrgyzstan us relations)	35.74
q: school mercury poisoning		10.29
$QDist(doc)$		16.64
0.2161	#combine(*school mercury poisoning*)	10.29
0.1284	#combine(school mercury exposure)	20.26
0.0441	#combine(school #1(mercury exposure))	13.64
0.0335	#combine(schools mercury poisoning)	9.11
q: blue grass music festival history		16.65
$QDist(doc)$		38.06
0.1953	#combine(*blue grass music festival history*)	16.65
0.1456	#combine(bluegrass music festival history)	50.10
0.0698	#combine(#1(bluegrass music) festival history)	23.90
0.0411	#combine(bluegrass #1(music festival) history)	22.46
q: kudzu pueraria lobata		44.96
$QDist(doc)$		51.83
0.2778	#combine(kudzu)	52.69
0.1244	#combine(*kudzu pueraria lobata*)	44.96
0.0596	#combine(#1(kudzu pueraria lobata))	22.08
0.0580	#combine(#1(kudzu kudzu))	1.93

Besides the original query, many reasonable and effective reformulated queries can also be observed in the learned query distribution. For example, "#1(cruise ship) damage sea life" and "kyrgyzstan #1(united states) relations" are segmented queries, where much better retrieval performance is achieved by discovering the concepts "cruise ship" and "united states" in original queries. "school mercury exposure" and "bluegrass music festival history" are substituted queries where the original query words "mercury poisoning" and "blue grass" are replaced with "mercury exposure" and "bluegrass", respectively. Significant performance improvement can be observed by using these substituted queries.

Furthermore, we present the query distribution learned by using both the document-level and the passage-level retrieval models QDist(doc+psg). Table 6.7 displays the example of this type of query distribution using the non-stemmed index.

Table 6.7 shows that besides using the document-level retrieval model, QDist(doc+psg) also applies the passage-level retrieval models to some important reformulated queries such as "#uw100(ephedra deaths)" and "#uw20(ephedra deaths)". These passage-level queries help further improve the performance of QDist(doc).

6.4 Results

The first experiment is conducted to compare the query distribution model with other reformulation models. RM and SDM represent the Concept Distribution (CDist) models. Seg-SVM generates a single segmented query and combines it with the original query, which can be considered as a variant of the Single Reformulated Query (SRQ) model. The results are provided in Table 6.8.

Table 6.8 shows that the two query distribution models outperform both the CDist models (SDM and RM) and the SRQ model (Seg-SVM), which supports the advantages of modeling reformulation as a distribution of queries instead of a distribution of concepts and a single reformulated query. Furthermore, QDist(doc+psg) outperforms

Table 6.7. Example of the query distribution learned on the non-stemmed index using the document-level and the passage-level retrieval models. Top ranked reformulated queries are displayed. In the query distribution, the original query is italicized. Average Precision (AP) is reported as the retrieval performance.

$P(q_r\|q)$ Reformulated Query (q_r)	AP
q: ephedra ma huang deaths	47.42
QDist(doc)	57.07
QDist(doc+psg)	62.58
0.1682 #combine(ephedra deaths)	61.78
0.1434 #combine(*ephedra ma huang deaths*)	47.42
0.0439 #combine(ephedra ma huang death)	47.12
0.0395 #combine(ephedra #1(ma huang) death)	47.41
0.0394 #combine(#1(ephedra sinica ma huang) deaths)	1.49
0.0392 #combine(ephedra sinica ma huang deaths)	28.83
0.0284 #uw100(ephedra deaths)	44.34
0.0276 #combine(#1(ephedra ephedra) deaths)	4.69
0.0233 #combine(#1(ephedra ma huang) death)	2.66
0.0193 #uw20(ephedra deaths)	36.45

QDist(doc), which shows that the dependencies imposed by the realistic queries are useful for both estimating the distribution and for retrieving the documents. Specifically, on Gov2, QDist(doc+psg) significantly outperforms SDM and RM in terms of both MAP and P10. SDM and RM are both strong baselines that represent the state-of-the-art techniques. The only exception happens on the Porter-stemmed index, where QDist(doc+psg) improves SDM by 1.9% on P10, which is not significant. On Robust04, QDist(doc+psg) significantly improves SDM and RM on MAP. Some improvement on P10 is also observed, but it is not significant. On ClueWeb, P10 is significantly improved by QDist(doc+psg), where the improvement over SDM and RM is as large as 15%-30%. Large improvements are also observed on MAP, but they are not significant. In addition, QL-psg is also worse than QDist methods, which shows that the benefits of QDist come from modeling the original query as a query distribution instead of simply using the passage information.

It is interesting to observe that the behavior of QDist(doc+psg) varies with different collections, although it generally achieves the best performance on all collections.

Table 6.8. The results of different reformulation models. The best performance is bolded. * denotes significantly different with baselines.

	Gov2		Robust04		ClueWeb	
	MAP	P10	MAP	P10	MAP	P10
Non-stemmed index						
QL	26.89	54.56	22.73	40.92	17.88	28.06
SDM	28.29	55.77	23.76	42.85	18.88	31.22
RM	28.72	56.04	24.82	42.21	18.18	28.88
Seg-SVM	27.66	56.71	23.38	41.49	18.30	30.71
QL-psg	26.52	54.43	22.99	41.41	16.73	27.35
QDist(doc)	31.09	60.87	25.76	**44.42**	20.23	35.92
vs. QL	*15.6%**	*11.6%**	*13.3%**	*8.6%**	*13.1%**	*28.0%**
vs. SDM	*9.9%**	*9.1%**	*8.4%**	*3.7%*	*7.2%*	*15.1%**
vs. RM	*8.3%**	*8.6%**	*3.8%*	*5.2%**	*11.3%*	*24.4%**
QDist(doc+psg)	**32.02**	**62.82**	**26.16**	44.10	**20.28**	**36.63**
vs. QL	*19.1%**	*15.1%**	*15.1%**	*7.8%**	*13.4%**	*30.5%**
vs. SDM	*13.2%**	*12.6%**	*10.1%**	*2.9%*	*7.4%*	*17.3%**
vs. RM	*11.5%**	*12.1%**	*5.4%**	*4.5%*	*11.6%*	*26.8%**
Porter-stemmed index						
QL	29.27	53.49	24.98	42.37	17.70	24.08
SDM	32.40	58.93	26.78	45.14	19.28	28.16
RM	31.07	54.83	26.71	42.85	18.03	25.00
Seg(SVM)	31.42	58.86	26.49	45.34	19.02	27.76
QL-psg	29.25	52.89	25.59	42.49	16.83	24.08
QDist(doc)	33.31	**60.67**	27.48	**45.50**	19.53	30.92
vs. QL	*13.8%**	*13.4%**	*10.0%**	*7.4%**	*10.3%*	*28.4%**
vs. SDM	*2.8%**	*3.0%*	*2.6%**	*0.8%*	*1.3%*	*9.8%*
vs. RM	*7.2%**	*10.7%**	*2.9%*	*6.2%**	*8.3%*	*23.7%**
QDist(doc+psg)	**33.98**	60.07	**27.88**	**45.50**	**20.13**	**32.55**
vs. QL	*16.1%**	*12.3%**	*11.6%**	*7.4%**	*13.7%**	*35.2%**
vs. SDM	*4.9%**	*1.9%*	*4.1%**	*0.8%*	*4.4%*	*15.6%**
vs. RM	*9.4%**	*9.6%**	*4.4%**	*6.2%**	*11.6%*	*30.2%**

One possible explanation for QDist(doc+psg)'s behavior is based on the properties of these three collections. Robust04 is a newswire collection containing high quality documents and its size is relatively small. On this collection, it is not difficult to retrieve some relevant documents using the original query, thus the baseline methods usually have good performance for the top end of the ranked list. Since RM and SDM already perform well on P10, it is not easy for QDist to significantly improve them. The effect of QDist is mainly shown on MAP (both SDM and RM are significantly improved), where the reformulated queries help retrieve more relevant documents that contain variations of the original query. In contrast, ClueWeb is a large web collection that contains significant "noise". QDist can significantly improve the quality of the top ranked documents (measured by P10) by considering evidence provided by the reformulated queries. In other words, QDist favors documents that are recommended by many reformulated queries not just the original query, which helps remove the noise. However, the improvement on MAP is not significant, since some relevant documents are difficult to retrieve in such a collection even using the reformulated queries. Gov2 is also a large web collection but the quality of web pages is higher than ClueWeb, thus it could be considered to be between Robust04 and ClueWeb in terms of its properties. QDist significantly improves MAP by retrieving more relevant documents and significantly improves P10 by increasing the quality of top-ranked documents.

Another observation is about the non-stemmed index and the Porter-stemmed index. In general, QDist provides more improvements over baselines using the non-stemmed index than using the Porter-stemmed one. It is not difficult to understand, since some of the effect of using the reformulated queries, especially morphologically similar queries is already provided by the Porter stemmer.

The second experiment is conducted to further analyze the query distribution model. Specifically, we consider two additional baselines. The first baseline is to use

Table 6.9. Further analysis of query distribution. QDist denotes QDist(doc).

	Gov2		Robust04		ClueWeb	
	MAP	P10	MAP	P10	MAP	P10
Non-stemmed index						
QL	26.89	54.56	22.73	40.92	17.88	28.06
SDM	28.29	55.77	23.76	42.85	18.88	31.22
RM	28.72	56.04	24.82	42.21	18.18	28.88
single	23.26	50.34	19.37	36.63	13.29	24.80
equal	28.21	57.52	24.62	42.73	17.40	31.22
QDist	31.09	60.87	25.76	44.42	20.23	35.92
Porter-stemmed index						
QL	29.27	53.49	24.98	42.37	17.70	24.08
SDM	32.40	58.93	26.78	45.14	19.28	28.16
RM	31.07	54.83	26.71	42.85	18.03	25.00
single	25.88	51.07	22.33	39.80	15.11	25.20
equal	31.31	58.86	26.55	44.46	16.97	30.00
QDist	33.31	60.67	27.48	45.50	19.53	30.92

the best reformulated query (except the original one) in the learned query distribution, which is denoted as "single". Compared with Seg-SVM, i.e., the variant SRQ method used in the previous experiment, "single" is an exact SRQ method and it uses the same set of reformulated queries as QDist. Thus, the comparison between "single" and QDist will focus on the effect of using the query distribution or a single best reformulated query. The second baseline assigns equal probabilities to all reformulated queries in the distribution, which is denoted as "equal". The comparison between "equal" and QDist helps show the effect of the probability estimation method used by QDist. Table 6.9 shows the results, where QDist(doc) is used as the query distribution method and QL, SDM and RM are used as references.

Table 6.9 shows that as with Seg-SVM, "single" is worse than QDist, which clearly indicates that using the whole query distribution is much better than using a single reformulated query. "single" is even worse than QL, which supports the observations of previous research (Bendersky and Croft 2008) that it is important to combine the original query and the reformulated query to achieve good performance on TREC

collections. "equal" is better than QL, comparable with SDM and RM, but it is still worse than QDist. This shows that the query probability estimation method proposed in this dissertation is effective. In addition, it is expensive to use the "equal" method, since it has to use all reformulated queries generated for retrieval. When the number of reformulated queries is big, this causes considerable computational cost. In contrast, QDist assigns appropriate probabilities to reformulated queries, thus it is easy to pick the top ranked reformulated queries for retrieval in practice. The effect of the number of reformulated queries chosen is explored in the third experiment.

In the third experiment, the effect of using the top ranked reformulated queries in the learned query distribution is shown in Fig. 6.1. QDist(doc) is used to represent the query distribution method. The results are reported using the non-stemmed index. Similar results are observed by using the Porter-stemmed index.

Fig. 6.1 shows that, on all collections, only using the top three to five reformulated queries can outperform both RM and SDM. The performance of using the top 15 reformulated queries is already very close to the performance of using the whole distribution.

The fourth experiment explores the effect of different types of features (PSG, NGRAM, OPER, see Table 6.2). We are particularly interested in their effects on picking the top ranked queries. The performance of the top three reformulated queries in QDist(doc) using different types of features is reported in Table 6.10. The best performance is bolded.

Table 6.10 shows that the best feature type is not consistent over collections. Generally, combining different types of features together is more effective than using a single type of feature. In addition to the top three reformulated queries, we also explore other numbers of top ranked reformulated queries. In general, when more reformulated queries are used, the performance of using different feature sets becomes more similar.

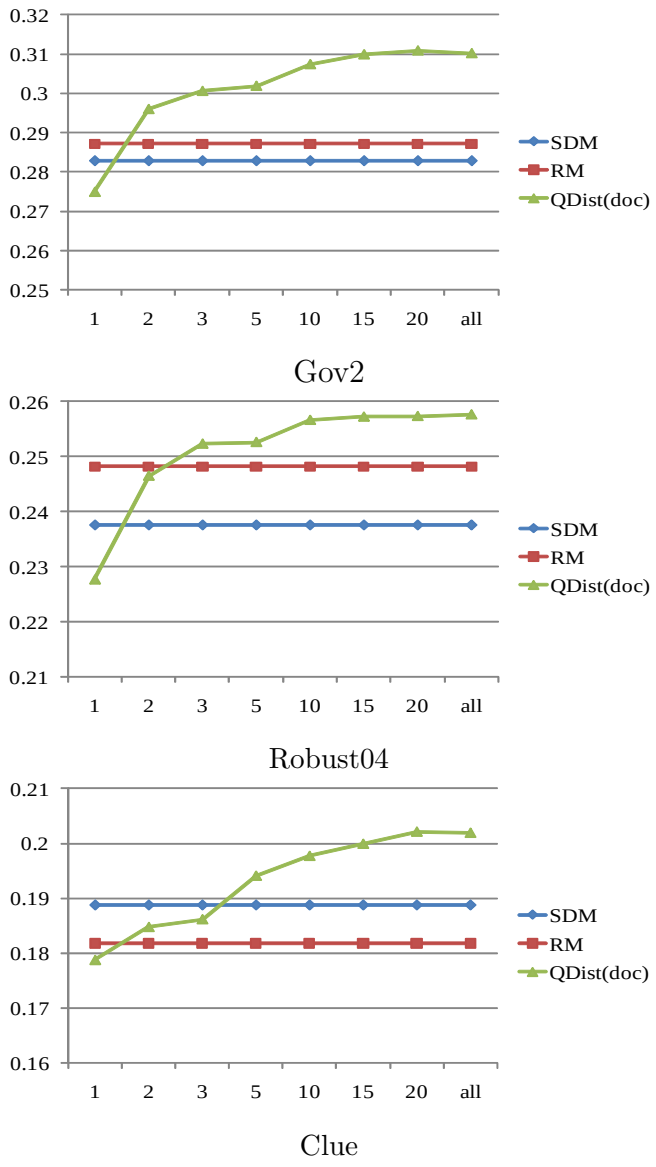


Figure 6.1. The effect of the number of reformulated queries. Non-stemmed index is used. x-axis is the number of top ranked reformulated queries and y-axis is MAP.

Table 6.10. The performance of top three queries in QDist(doc) using different types of features. p denotes PSG, n denotes NGRAM and o denotes OPER.

	Gov2		Robust04		ClueWeb	
	MAP	P10	MAP	P10	MAP	P10
Non-stemmed index						
p	**30.36**	58.39	24.52	42.49	19.18	30.82
n	29.45	57.58	24.93	43.45	19.40	32.04
o	27.86	55.64	24.67	42.53	18.81	31.43
p+n	30.29	**59.19**	25.16	**43.98**	**19.57**	**33.98**
p+o	30.14	58.19	24.61	42.81	18.43	31.53
n+o	28.63	56.38	25.22	43.82	18.60	31.22
all	30.06	57.85	**25.23**	43.82	18.62	33.06
Porter-stemmed index						
p	31.94	56.91	25.90	43.21	18.60	28.27
n	31.93	**57.38**	26.64	44.62	19.14	29.69
o	30.98	56.31	26.75	44.54	19.11	29.59
p+n	32.02	56.64	26.73	44.50	19.07	**30.31**
p+o	32.06	**57.38**	26.59	44.06	18.86	29.29
n+o	31.36	56.38	27.00	**44.78**	**19.19**	29.39
all	**32.10**	57.11	**27.01**	**44.78**	18.91	29.80

6.5 Summary

In this chapter, we described an implementation of the query distribution model for short queries. Experiments on TREC collections have shown that the query distribution model significantly outperforms the state-of-the-art techniques.

CHAPTER 7

APPLICATION II: QUERY DISTRIBUTION FOR VERBOSE QUERIES

Verbose (or long) queries have become popular in *ad hoc* retrieval recently, since they help users express their information needs naturally. However, current search engines cannot handle verbose queries well. It seems that the additional information provided in verbose queries is more likely to confuse current search engines rather than help them. Thus, dealing with verbose queries poses a new challenge for *ad hoc* retrieval.

Previous work on processing verbose queries generally falls into two areas, selecting a subset of the verbose query and weighting all query words of the verbose query.

Kumaran and Allan (2007) studied reducing the verbose query into a subset through human interaction. The subset queries with high mutual information scores are displayed to users. Their experiments show that a user can select good subset queries using snippet information. Bendersky and Croft (2008) proposed a method to find key concepts from a verbose query using different types of features. A key concept can be considered as a special subset query, which improves the retrieval performance when combined with the verbose query. Kumaran and Carvalho (2009) used Ranking SVM to learn selecting sub-queries using several query quality predictors. Ranking SVM learns to rank subset queries according to their retrieval performance.

Lease et al (2009) improved verbose queries by weighting query terms, which assigned more weight to important words and less weight to unimportant ones. They trained a regression model to learn how to map the secondary features to the optimal

Table 7.1. Implementation of the Query Distribution Model for verbose queries.

Query Generation	subset selection (Section 4.2)
Probability Estimation	optimizing a single query (Section 5.2)
Retrieval Model	four types of retrieval models (Section 7.1.2)

weights of query words. Lease (2009) further incorporated their regression model into the framework of the sequential dependence model (Metzler and Croft 2005) and observed significant performance improvement. Bendersky et al (2010) proposed a unified framework to measure the importance of concepts underlying the verbose queries and extended the conventional sequential dependence model to a weighted version.

In this chapter, we describe how to apply the query distribution model for verbose queries. The content of this chapter has been published in our previous work (Xue *et al.* 2010).

7.1 Implementation of Query Distribution Model

In order to implement the Query Distribution Model for verbose queries, we need to instantiate three components, i.e., query generation, probability estimation and retrieval model. For query generation, we use the subset selection technique described in Section 4.2. For probability estimation, we optimize the performance of a single reformulated query as shown in Section 5.2. For the retrieval model, four types of retrieval models are developed. The query features used for probability estimation and the retrieval models will be described in the rest of this section. The above choices are summarized in 7.1.

7.1.1 Query Features

Three types of features are used to characterize a reformulated query q_r (or a subset query in this chapter). Some of the features used here are the same as the features used by Bendersky and Croft (2010) and by Kumaran and Carvalho (2009).

Independency Features (ID) characterize a single query word. This type of feature includes the standard term frequency and document frequency in the target corpus, the frequency of the query word observed in external resources such as Google Ngram, Wikipedia and some commercial query logs. Besides statistical information, they also include some syntactic features such as POS-tags. For a given independency feature, the feature value of each word is summed together to characterize q_r.

Local Dependency Features (LD) capture the dependencies between query words. Since they characterize *some* query words but not all, they are called *local* dependency features. This type of feature includes bigrams, noun phrases, the dependency relations returned by a dependency parser (De Marneffe and Manning 2008) and named entities. All statistical features used for single words can also be applied to bigrams. Noun phrases have been shown to be effective when they are combined with the original query (Bendersky and Croft 2008). A dependency parser (De Marneffe and Manning 2008) can return different types of relations between two query words. For example, given the sentence "How frequently does the Mississippi River flood its banks?", "river" and "flood" satisfies the subject relation, since "river" is the subject of "flood". Also, "flood" and "banks" have the object relation, since "banks" is the object of "flood". It helps sometimes to include words satisfying a certain relation in a subset query. Named entities including names of people, locations and organizations are usually important and may be included in subset queries. For a local dependency feature, the feature values of all local dependencies are summed together to characterize q_r.

Global Dependency Features (GD) describe a subset query as a whole. Thus, they are called *global* dependency features. As indicated by Kumaran and Carvalho (2009), query quality predictors are good features to characterize subset queries. Here, several types of query quality predictors have been used, such as Mutual Information (Kumaran and Allan 2007), Query Scope (He and Ounis 2004), Query Clarity

(Cronen-Townsend *et al.* 2002) and so on. Passage-level evidence can also be used to describe a subset query. If a subset query appears frequently within a passage, it is very likely that query words of this subset query are closely related. Thus, the number of passages containing a subset query is used as a feature. The parsing tree (Klein and Manning 2003) of the input query also provides valuable information. It is interesting to consider whether query words of a given subset query concentrate on a small part of the parsing tree or spread over the whole tree. This property is partly measured by the height of the lowest common ancestor of all query words in the parsing tree.

Finally, all features used are summarized in Table 7.2.

7.1.2 Retrieval Model

A retrieval model takes into a reformulated (or subset) query q_r and outputs the retrieval score. Four types of retrieval models are developed as follows.

The first two types of models only use q_r, where **SubQL** uses the query likelihood language model and **SubSDM** uses the sequential dependence model. Here, "Sub" is used to emphasize that it is used for a subset query.

The other two types of retrieval models combine the subset query q_r with the original query q.

QL+SubQL denotes a combination of the original query and the subset query, where both parts use the query likelihood language model. The score of a document using this model can be calculated as follows:

$$score(D, q, q_r) = \alpha score_{QL}(D, q) + (1 - \alpha) score_{QL}(D, q_r) \tag{7.1}$$

where $score_{QL}(D, q)$ is the retrieval score of using the query likelihood language model and α is a parameter weighting the original query and the subset query. α is set as 0.8 in this chapter.

Table 7.2. Three types of features

Independency Features (ID)	
uTF	unigram term frequency
uDF	unigram document frequency
uNGram	unigram count in Google nGram
uWiki	unigram count of matching Wiki titles
uMSNLog	unigram count in MSN query logs
uPosTag	unigram pos-tag="NN", "VB", "JJ"
Local Dependency Features (LD)	
bTF	bigram term frequency
bDF	bigram document frequency
bNGram	bigram count in Google nGram
bWiki	bigram count of matching Wiki titles
bMSNLog	bigram count in MSN query logs
np	noun phrases
dep-obj	the object relation
dep-subj	the subject relation
dep-nn	the noun compound modifier relation
PER	person names
LOC	location names
ORG	organization names
Global Dependency Features (GD)	
MI	mutual information
SQLen	sub-query length
QS	query scope
QC	query clarity score
SOQ	similarity to original query
psg	count of passages containing sub-query
h-pnode	height of the parent node covering sub-query

Table 7.3. Example of Indri queries.

```
q : jobs outsourced india
q_s: jobs india
SubQL:
        #combine(jobs india)
SubSDM:
        #weight(
          0.85 #combine(jobs india)
          0.1  #combine(#1(jobs india))
          0.05 #combine(#uw8(jobs india))
        )
QL+SubQL:
        #weight(
          0.8 #combine(jobs outsourced india)
          0.2 #combine(jobs india)
        )
SDM+SubQL:
        #weight(
          0.8 #wegiht(
              0.85 #combine(jobs outsourced india)
              0.1 #combine(#1(jobs outsourced) #1(outsourced india))
             0.05 #combine(#uw8(jobs outsourced)  #uw8(outsourced india))
          )
          0.2 #combine(jobs india)
        )
```

SDM+SubQL uses the sequential dependency model for the original query and uses the query likelihood model for the subset query. The score of a document is calculated as follows:

$$score(D, q, q_r) = \alpha score_{SDM}(D, q) + (1 - \alpha) score_{QL}(D, q_r) \tag{7.2}$$

where $score_{SDM}$ is the retrieval score of using a sequential dependence model and α is a parameter. α is set as 0.8 in this chapter.

The above four retrieval models can all be implemented using the Indri query language (Metzler and Croft 2004). Table 7.3 shows an example of Indri queries used for each retrieval model.

7.2 Experimental Configuration

Experiments are conducted on three TREC collections (Gov2, Robust04 and WT10g). The details of these collections can be found in the Appendices.

For each collection, the index is built using the Porter Stemmer (Porter 1980). No stopword removal is performed during indexing. For each topic, the *description* part is used as the query. Following Bendersky et al (Bendersky *et al.* 2010), a short list of 35 stopwords and some frequent stop patterns (e.g., "find information") are removed from the description query in order to improve the retrieval performance of the baseline methods.

The query set is split into a training set and a test set. On the training set, the parameters λ_k are learned according to Section 5.2. On the test set, the learned parameters λ_k are used to generate the subset distribution for each test query and the performance of using the generated subset distribution is reported. Ten-fold cross validation is conducted.

Several baseline methods are compared. Besides the query likelihood language model (QL) and the sequential dependence model (SDM), we also include Kumaran and Carvalho's method (Kumaran and Carvalho 2009), which considers sub-query selection as a ranking problem. This method is denoted as SRank. Rank SVM (Joachims 2002) is used as the ranking model. According to their suggestions, the parameters of Rank SVM are set as follows: RBF kernel is used with γ set as 0.001 and C is set as 0.01. Another baseline is Bendersky and Croft's method (Bendersky and Croft 2008) that augments the original query by discovering key concepts. This method is denoted as KC. The baselines are summarized in Table 7.4.

The standard performance measures, mean average precision (MAP) and precision at 10 (P@10), are used to measure retrieval performance[1]. The details of the perfor-

[1]In order to improve readability, we report 100 times the actual values of each performance measure.

Table 7.4. Summary of Baselines

Name	Description
QL	query likelihood language model
SDM	sequential dependency model
SRank	Rank SVM
KC	discovering key concept

mance measures can be found in the Appendices. The two-tailed t-test is conducted for significance.

7.3 Examples

First, we present some examples of the subset distribution in Table 7.5. The average precision (AP) of the original query and subset queries is reported on the Gov2 collection. As mentioned previously, some stopwords and stop patterns are removed from the original query. Those words are kept to improve readability. Note that they are not used for retrieval and sub-query generation.

Table 7.5 shows that the subset distribution learned is reasonable, which successfully assigns high probabilities to subset queries that perform better than the original query. For example, given the original query "steps manage control protect squirrels", the top three subset queries "steps protect squirrels", "steps control squirrels" and "steps control protect squirrels" receive most of the probability and all of them perform much better than the original query.

Furthermore, it is interesting to compare the subset distributions learned based on different retrieval models. Table 7.6 compares the top one subset query returned by SubQL and QL+SubQL and some examples are also provided to compare SubDM with SDM+SubQL.

Table 7.6 shows that when the subset queries are used alone (SubQL and SubDM), the subset distribution learned tends to assign high probabilities to longer subset queries, since it is safer to cover most of the concepts in the original query. When the

Table 7.5. Example of the subset distribution learned using SubQL. Top four subset queries are displayed. In the original query q, the words actually used are bolded and stopwords and stop structures are italicized.

$P(q_r\|q)$	subset query (q_r)	AP
q: *what is the* **history** *and* **location** *of* **scottish highland games** *in the* **united states**		49.53
0.564	scottish highland games united states	54.28
0.341	location scottish highland games united states	57.04
0.074	history scottish highland games united states	43.73
0.010	history location scottish highland united states	18.38
q: *give information on* **steps** *to* **manage control** *or* **protect squirrels**		21.03
0.621	steps protect squirrels	31.13
0.324	steps control squirrels	28.59
0.048	steps control protect squirrels	30.35
0.002	steps manage squirrels	25.46
q: **how** *have* **humans responded** *and* **how should they respond** *to the* **appearance** *of* **coyotes** *in* **urban** *and* **suburban areas**		10.47
0.452	humans responded appearance coyotes urban suburban	31.54
0.103	humans responded respond appearance coyotes urban	26.90
0.086	humans responded respond appearance coyotes suburban	14.09
0.045	humans responded appearance coyotes urban	50.80
q: *what is* **known about** *the* **culture** *and* **history** *of the* **chaco people** *from* **features** *of the* **chaco culture national historic park**		30.77
0.214	culture history chaco people features chaco	53.16
0.119	culture chaco people features chaco park	46.98
0.102	known chaco people features chaco park	38.60
0.069	history chaco people features chaco park	42.01
q: *the* **remedies** *and* **treatments given** *to* **lessen** *or* **stop effects** *of* **ovarian cancer**		13.53
0.835	remedies treatments given effects ovarian cancer	23.92
0.073	remedies treatments given lessen ovarian cancer	17.31
0.066	remedies treatments given ovarian cancer	20.37
0.010	remedies treatments lessen effects ovarian cancer	16.79

Table 7.6. Comparisons of the top subset queries in the subset distributions learned using different retrieval models. In the original query q, the words actually used are bolded and stopwords and stop structures are italicized.

q: *what* **evidence** *is* **there** *that* **aspirin may help prevent cancer**
SubQL: evidence aspirin may help cancer QL+SubQL: aspirin prevent cancer
q: *what* **states** *or* **localities offer programs** *for* **gifted talented students**
SubQL: localities offer programs gifted talented students QL+SubQL: gifted talented students
q: **illicit activity involving diamonds** *to* **include diamond smuggling**
SubQL: illicit activity diamonds diamond smuggling QL+SubQL: diamonds diamond smuggling
q: *what* **allegations** *have been* **made about enrons culpability** *in the* **california energy crisis**
SubSDM: allegations enrons culpability california energy crisis SDM+SubQL: enrons culpability california energy crisis
q: *what is the* **history** *and* **location** *of* **scottish highland games** *in the* **united states**
SubSDM: location scottish highland games united states SDM+SubQL: scottish highland games
q: **where** *do* **yew trees grow anywhere** *on the* **globe**
SubSDM: yew trees grow globe SDM+SubQL: yew trees globe

subset queries are combined with the original query (QL+SubQL, SDM+SubQL), the subset distribution learned tends to favor shorter queries, since it is reasonable to focus on important concepts when the original query has covered all concepts. For example, given the original query "history location scottish highland games united states", SubSDM selects a sub-query "location scottish highland games united states" which covers almost all concepts of the original query, while SDM+SubQL simply picks up the most important concept "scottish highland games". Similarly, SubQL selects a subset query "localities offer programs gifted talented students" which only removes "states" from the original query, while QL+SubQL selects the key concept "gifted talented student".

Table 7.7. Performance of retrieval models using the subset distribution.^{q} denotes significantly different with QL and $_d$ denotes significantly different with SDM.

	Gov2		Robust04		Wt10g	
	MAP	P@10	MAP	P@10	MAP	P@10
QL	25.43	52.21	25.49	43.13	19.61	32.68
SDM	27.85	54.03	26.83	44.94	20.87	**35.77**
SRank	24.99	50.74	24.78	41.57	19.98	32.06
KC	27.52	53.83	25.97	41.65	21.01	34.02
QDist Methods						
SubQL	26.66^q	53.36	25.96	41.93	19.27	31.75
QL+SubQL	26.76^q	53.15	26.20^q	43.21	19.94	33.20
SubSDM	28.60^q	53.76	27.07^q	43.69	20.70	34.74
SDM+SubQL	$\mathbf{28.70}^q_d$	55.37^q	$\mathbf{27.37}^q_d$	45.14^q	$\mathbf{22.17}^q$	35.15^q

7.4 Results

The first experiment is conducted to compare the subset distribution models with baseline methods. The models using the subset distribution include SubQL, QL+SubQL, SubSDM and SDM+SubDM. For each model, the top ten subset queries within the subset distribution are kept. The results are shown in Table 7.7. The best performance of each column is bolded.

Table 7.7 shows that SubQL performs better than QL on Gov2. QL+SubQL performs better than QL on all three collections, which indicates that combining the sub-query with the original query is promising. SubSDM is slightly better than SDM on Gov2 and Robust04 according to MAP. The best performance is achieved by SDM+SubQL, which performs significantly better than SDM, a very strong baseline, on Gov2 and Robust04. It indicates that the most effective way is to use the sequential dependence model for the original query, use the query likelihood model for the sub-query and combine them together. SDM+SubQL also performs better than KC, the state-of-the-art technique for improving verbose queries.

The second experiment is conducted to explore the relative effect of different types of features (Independency Features, Local Dependency Features and Global Depen-

dency Features). Table 7.8 reports the mean average precision (MAP) of different retrieval models using different types of features.

Table 7.8 shows that the best feature set is not consistent over different retrieval models and different collections. On Gov2, different retrieval models prefer different features. On Robust04, combining all three types of features together is clearly the best choice. On Wt10g, only using the Independency Features and the Global Dependency Features (I+G) seems to be the best choice. It is also interesting to notice that the best feature set almost always involves Global Dependency Features, which shows that capturing global dependencies of queries is important. Generally, combining different types of features together is more effective than only using a single type of features.

Table 7.7 has shown the performance of using the top ten subset queries. In the third experiment, we explore the effect of the number of subset queries used. The results are displayed in Fig. 7.1. The retrieval model used is SubQL.

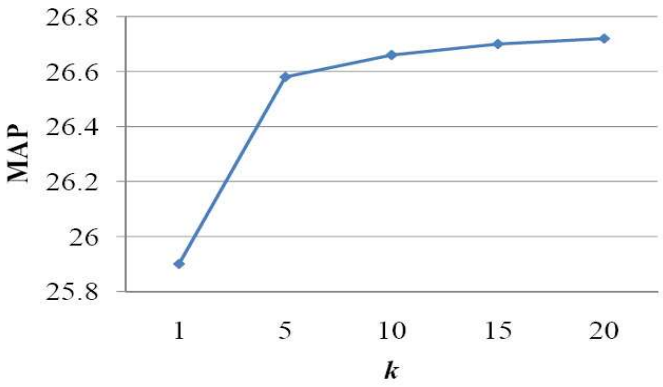
Fig. 7.1 shows that on three collections the most obvious performance improvement happens when k increases from one to five and there is not much difference when k is bigger than five. The reason is that the top five sub-queries usually receive most of the probability of the distribution. The above observations support using a distribution of sub-queries for improving retrieval performance. Thus, instead of returning a single sub-query, the model that can generate the sub-query distribution is preferred.

7.5 Summary

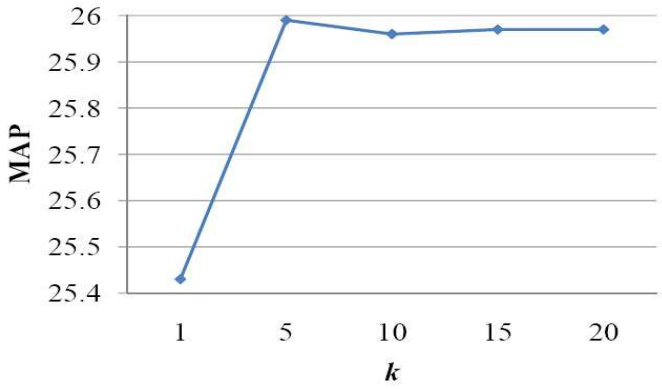
In this chapter, we described an implementation of the query distribution model for verbose queries. Experiments on TREC collections have shown that the query distribution model significantly outperforms the state-of-the-art techniques.

Table 7.8. The mean average precision (MAP) of using different types of features. "I" denotes the Independency Features, "L" denotes the Local Dependency Features and "G" denotes the Global Dependency Features.

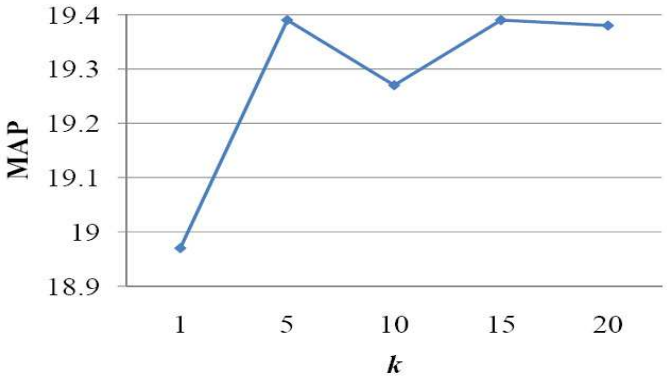
	SubQL	QL+SubQL	SubSDM	SDM+SubQL
	Gov2			
I	26.20	26.55	**29.04**	28.71
L	26.27	26.32	28.57	28.54
G	25.78	26.19	28.30	28.49
I+L	26.02	26.78	28.23	28.62
I+G	26.49	26.62	28.84	**28.96**
L+G	26.57	**26.90**	28.79	28.60
I+L+G	**26.66**	26.76	28.60	28.70
	Robust04			
I	25.53	25.63	26.77	26.86
L	25.04	25.88	26.44	27.25
G	25.69	25.81	26.87	26.99
I+L	25.46	26.08	26.70	27.32
I+G	25.77	26.12	26.75	27.25
L+G	25.59	26.19	26.44	27.06
I+L+G	**25.96**	**26.20**	**27.07**	**27.37**
	Wt10g			
I	21.00	20.34	22.19	22.73
L	19.33	20.10	20.51	21.55
G	21.15	20.11	22.14	21.98
I+L	19.54	19.92	20.55	21.96
I+G	**21.84**	**21.11**	**23.21**	**22.76**
L+G	19.38	19.99	20.59	22.47
I+L+G	19.27	19.94	20.70	22.17



(a) Gov2



(b) Robust04



(c) Wt10g

Figure 7.1. The influence of the number of subset queries

CHAPTER 8

HIERARCHICAL QUERY DISTRIBUTION

Dealing with complex queries usually requires a series of query operations. For example, a reasonable process of dealing with a verbose query can be described as follows. The system first selects a subset of query words from the original query to remove noisy information. Then, the generated subset query is further modified to handle vocabulary mismatch. Finally, weights are assigned to queries generated at each step. Depending on the application, the above process could become more complicated. For example, in cross-lingual retrieval, the original verbose query needs to be translated into a foreign language query before applying any further operation. The above process will generate multiple sequences of reformulated queries, where each sequence records a way of modifying the original query using several query operations. These reformulation sequences capture the relationships between the reformulated queries. Fig. 8 displays some examples of the reformulation sequences, where the subset query is selected from the original query at the first step of the sequence and the second step further substitutes the subset query.

The query distribution model proposed earlier in this dissertation indeed considers a reformulated query as the basic unit, but it fails to capture the relationships between

Reformulation Sequence
$Q \rightarrow$reductions iraqs foreign debt$\rightarrow$reduce iraqs foreign debt
$Q \rightarrow$iraqs foreign debt$\rightarrow$iraqs foreign debts
$Q \rightarrow$iraqs foreign debt$\rightarrow$iraqs external debt

Figure 8.1. The reformulation sequences generated for the verbose query q "any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt"

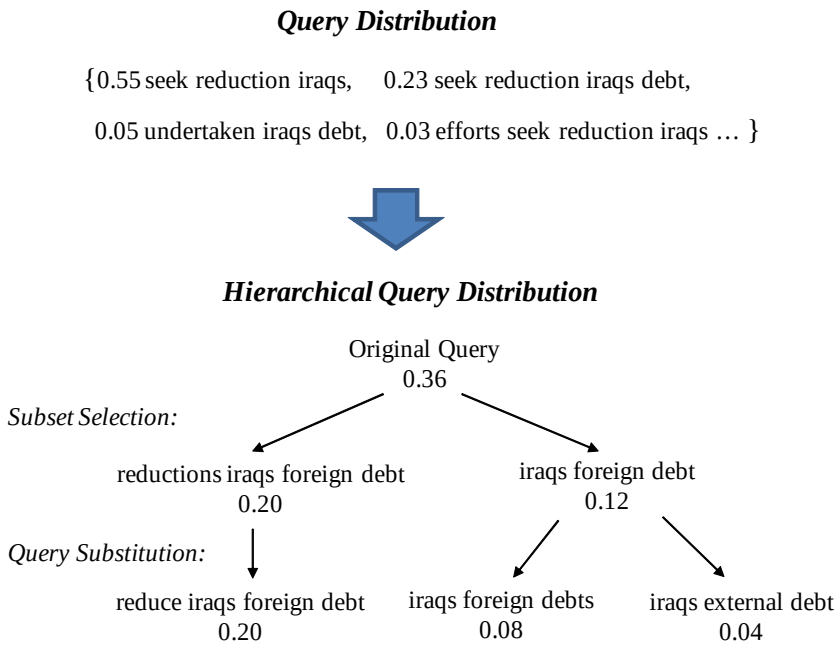


Figure 8.2. From the query distribution to the hierarchical query distribution for the verbose query "identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt"

the reformulated queries. Thus, the sequences of reformulated queries cannot be modeled. In this chapter, we propose a hierarchical query distribution model that extends the standard query distribution model to capture the dependence between reformulated queries. Using the hierarchical query distribution, a query is transformed into a reformulation tree, where the nodes at each level of this tree correspond to the reformulated queries generated using a specific query operation. A reformulation sequence is naturally modeled as a path from the root node to the leaf node. The construction of the reformulation tree simulates the process of applying a series of query operations to the complex query.

Using a verbose query as an example, we implement the hierarchical query distribution framework as a two-level tree structure, where the first level corresponds to the subset query selection operation and the second level corresponds to the query substitution operation.

Fig. 8.2 illustrates the hierarchical query distribution. The first level of this tree consists of two subset queries extracted from the original query, i.e., "reductions iraqs foreign debt" and "iraqs foreign debt". At the second level, each subset query is further modified to generate query substitutions. For example, "iraqs foreign debt" has been modified to "iraqs external debt". Furthermore, the probability is assigned to each node of this tree, which measures the importance of each reformulated query.

The content of this chapter has been published in our previous work (Xue and Croft 2012).

8.1 Framework

Suppose that n query operations $\{r_1, r_2, ..., r_n\}$ are required to process a complex query q. Then, q is transformed into a n-level tree T. Each node of T represents a reformulated query q_r. From now on, if not explicitly indicated, we use q_r to represent both a node of T and the corresponding reformulated query. The root node of T represents the original query q, which can be considered as a special reformulated query. The ith level of T are generated by applying the ith operation r_i to the nodes at the $(i-1)$th level. An arc is added between the nodes at the $(i-1)$th level and the nodes at the ith level if the latter is the output of applying r_i to the former. Therefore, each path of T corresponds to a reformulation sequence. Furthermore, the probability $P(q_r|q)$ is assigned to each node of T, which measures the importance of the corresponding reformulated query q_r. Since the sum of the probability assigned to each reformulated query is equal to one, i.e., $\sum_{q_r \in T} P(q_r|q) = 1$, $P(q_r|q)$ is considered as a multinomial distribution. Furthermore, all reformulated queries are organized into a hierarchial structure, thus this query distribution is called the hierarchical query distribution.

When a series of query operations are applied, this representation helps model the reformulation sequences generated using these operations, which cannot be captured

by the query distribution model. Similar to the query distribution model (as shown in Fig. 3.1), the framework of the hierarchical query distribution also consists of three major components.

First, the reformulated queries are generated by applying a series of reformulation operations. The reformulated queries are further organized into a tree structure, which captures different reformulation sequences.

Second, the probability is estimated for each node (or reformulated query) in the tree. This probability should characterize both the properties of this node itself and the relationships with other nodes. Specifically, the probability of q_r is not only decided by its own query features $\{f_k\}$ but also by the weight of its parent node $par(q_r)$. In this way, the relationships between reformulated queries are incorporated into the probability estimation.

Third, when the reformulation tree T, the retrieval score of a document D is calculated using Eq. 8.1.

$$score(q, D) = \sum_{q_r \in T} P(q_r|q) \log P(q_r|D) \tag{8.1}$$

Eq. 8.1 is the same as the retrieval score of using the query distribution as shown in Eq. 3.1, where the retrieval score is a linear combination of using each reformulated query.

8.2 Hierarchical Query Distribution for Verbose Queries

In this section, we instantiate the general hierarchical query distribution for verbose queries and describe a two-level reformulation tree. We first describe the query operations used to construct the reformulation tree, i.e. subset query selection and query substitution. Then, we introduce a stage-based weight estimation method to assign weight to each node. These weights are finally transformed into probabilities.

8.2.1 Building Tree Structure

The construction of the reformulation tree for verbose queries consists of two steps: first, subset queries are selected from the original query; second, the subset queries generated in the previous step are further modified to generate query substitutions.

The subset query selection is described in Section 4.2. We follow Kumaran and Carvalho (2009)'s method to generate subset queries. All query words from the original verbose query are considered. If the length of the verbose query is bigger than ten, we first rank all query words by their *idf* values and then pick the top ten words for the subset query generation. Then, all subset queries with length between three and six words are generated.

The passage analysis technique is used to generate query substitutions, where the details can be found in Section 4.1. Briefly, in order to replace one word from the original query, all the passages containing the rest of the query words are first extracted. Then, the three methods introduced in Section 4.1 are used to generate candidates for query substitution from these passages. **Morph** considers the morphologically similar words as candidates. **Pattern** considers the words matching the patterns extracted from the original query as candidates. **Wiki** considers the Wikipedia redirect pairs as the candidates. Finally, the top ranked candidates are used as query substitutions.

Given the above two query operations, the reformulation tree for the verbose query can be generated in this way. First, all subset queries with length between three to six are extracted from the original query. Each subset query is assigned a weight. How to estimate the weight will be described later. According to this weight, we will pick the top ranked subset queries to construct the first level of the reformulation tree. *SubNum* is a parameter that controls the number of top ranked subset queries used for the first level construction. Second, among these *SubNum* subset queries, we further modify the top *ModNum* queries to generate query substitutions, which

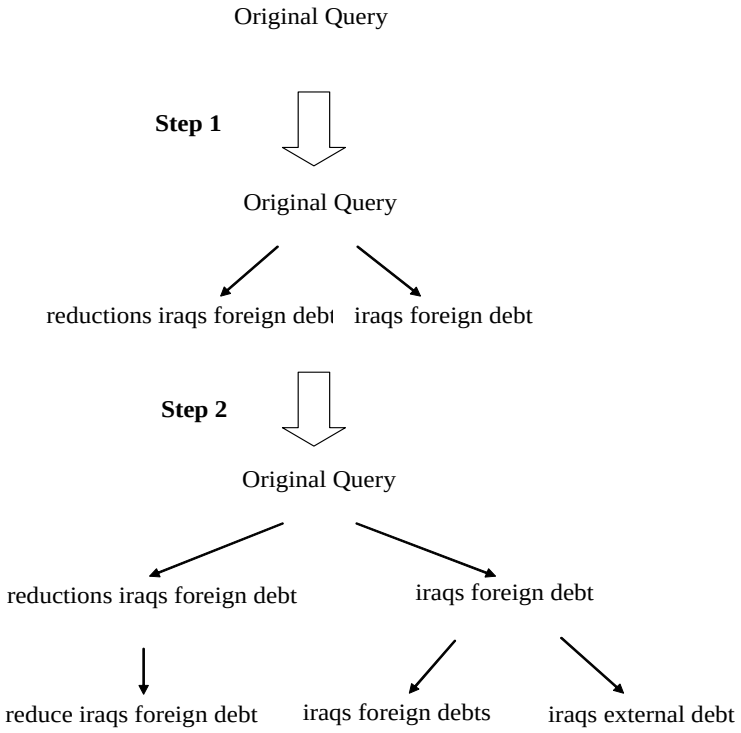


Figure 8.3. The process of constructing a reformulation tree

constructs the second level of the reformulation tree. *ModNum* is another parameter that controls the number of the top ranked subset queries used for query substitution.

For example, the reformulation tree displayed in Fig. 8.2 can be constructed in two steps. This process is illustrated in Fig. 8.3. First, we pick the top two subset queries "reductions iraqs foreign debt" and "iraqs foreign debt" to construct the first level of the reformulation tree. Second, we modify these two subset queries respectively. For the first subset query, "reduce iraqs foreign debt" is generated by replacing "reduction" with "reduce". For the second subset query, two query substitutions, i.e. "iraqs foreign debt" and "iraqs external debt" are generated.

8.2.2 Weight Estimation

Similar to Section 5.3, we use $w(q_r)$ to replace $P(q_r|q)$ for learning the parameters. $w(q_r)$ can be calculated in Eq. 8.2.

$$w(q_r) = w(par(q_r)) \sum_k \lambda_k f_k(q_r) \tag{8.2}$$

Eq. 8.2 indicates that the weight of a node in the reformulation tree depends on both its intrinsic features and the weight of its parent nodes. In this part, we describe a stage-based weight estimation method.

In the initial stage, the root node (the original query q) is assigned the weight 1, i.e. $w(q) = 1$.

In **Stage I**, after the subset queries q_{sub} are generated, we calculate the weight of q_{sub} using Eq. 8.3.

$$\begin{aligned} w(q_{sub}) &= w(q) \sum_k \lambda_k^{sub} f_k^{sub}(q_{sub}) \\ &= \sum_k \lambda_k^{sub} f_k^{sub}(q_{sub}) \end{aligned} \tag{8.3}$$

Eq. 8.3 instantiates Eq. 8.2 by focusing on the subset queries. f_k^{sub} is the query feature extracted from q_{sub} and λ_k^{sub} is the corresponding parameter. Since the root node is the parent of every subset query, its weight $w(q) = 1$, is used in Eq. 8.3.

In order to estimate $\{\lambda_k^{sub}\}$ by directly optimizing the retrieval performance, we use the same approach described in Section 5.3. Specifically, we transform each query feature f_k^{sub} into the corresponding retrieval feature F_k^{sub}, where F_k^{sub} is calculated in Eq. 8.4.

$$F_k^{sub}(\{q_{sub}\}, D) = \sum_{q_{sub}} f_k^{sub}(q_{sub}) \log P(q_{sub}|D) \tag{8.4}$$

where $\log P(q_{sub}|D)$ is the retrieval score of using q_{sub} to retrieve D. The calculation of $\log P(q_{sub}|D)$ depends on the retrieval model. The retrieval feature F_k^{sub} combines the retrieval score of each subset query q_{sub} using their corresponding query feature $f_k^{sub}(q_{sub})$ as the combination weight. In general, F_k^{sub} indicates how well documents are ranked if f_k^{sub} is used as the weight to combine subset queries.

Now, we obtain a set of retrieval features $\{F_k^{sub}\}$. The problem of estimating $\{\lambda_k^{sub}\}$ to combine the query features $\{f_k^{sub}\}$ is transformed into the problem of combining the corresponding retrieval features $\{F_k^{sub}\}$ to achieve the best retrieval performance. The latter problem is typically solved using learning to rank techniques. Here, the ListNet method (Cao *et al.* 2007) is adopted to learn $\{\lambda_k^{sub}\}$ on the training set.

After obtaining $\{\lambda_k^{sub}\}$, we can assign the weight for each subset query according to Eq. 8.3.

In **Stage II**, we assign weights to the substituted queries. The weight of a substituted query q_{mod} is calculated using Eq. 8.5.

$$w(q_{mod}) \quad = \quad w(q_{sub}) \sum_k \lambda_k^{mod} f_k^{mod}(q_{mod}) \tag{8.5}$$

where q_{sub} is the parent node of q_{mod}. Compared with Eq. 8.3, the weights of the subset queries $w(q_{sub})$ generated in Stage I are incorporated in Eq. 8.5.

Similarly, in order to estimate $\{\lambda_k^{mod}\}$, we transform f_k^{mod} to the corresponding retrieval feature F_k^{mod} using Eq. 8.6.

$$F_k^{mod}(\{q_{mod}\}, D) = \sum_{q_{mod}} w(q_{sub}) f_k^{mod}(q_{mod}) \log P(q_{mod}|D) \tag{8.6}$$

where q_{sub} is the parent node of q_{mod}. In general, F_k^{mod} tells how well the documents are ranked if f_k^{mod} is used as the weight to combine the substituted queries. Thus, the parameters $\{\lambda_k^{mod}\}$ are learned by combining these retrieval features $\{F_k^{mod}\}$ using ListNet.

Finally, we normalize the weights of every nodes in the reformulation tree to make them sum to one.

Table 8.1. Summary of features

Features for subset selection	
MI	mutual information
SQLen	sub-query length
QS	query scope
QC	query clarity score
SOQ	similarity to original query
psg	count of passages containing q_{sub}
h-pnode	height of the parent node covering sub-query
KeyCpt	whether contains the key concept
Features for query substitution	
Morph	generated using Morph
Pattern	generated using Pattern
Wiki	generated using Wiki
psg	count of passages containing q_{mod}
seg-type	the number of possible segmentations

8.2.3 Query Features

In this part, we describe the query features used to characterize the subset queries and the substituted queries.

The features used to characterize the subset queries are mainly the Global Dependency Features described in Table 7.2. In addition, whether a subset query contains key concepts is also considered as a feature. These key concepts were discovered using the method proposed by Bendersky and Croft (2008).

The features used to characterize the substituted queries are borrowed from Table 6.2, which are the type of the substitution methods and the passage information. Furthermore, the number of possible segmentations of a substituted query is used as another feature. This feature can be directly obtained using the passage analysis technique as described in Section 4.1.

The details of the features are summarized in Table 8.1.

8.2.4 Retrieval Model

The retrieval score of using a reformulation tree T can be calculated using Eq. 8.1. For example, given the reformulation tree shown in Fig. 8.2, the retrieval score can be calculated as follows:

$$\begin{aligned} sc(T, D) &= 0.36 \times sc(\text{Original Query}, D) \\ &= +0.2 \times sc(\text{“reductions iraqs foreign debt”}, D) \\ &= +0.12 \times sc(\text{“iraqs foreign debt”}, D) \\ &= +0.2 \times sc(\text{“reduce iraqs foreign debt”}, D) \\ &= +0.08 \times sc(\text{“iraqs foreign debts”}, D) \\ &= +0.04 \times sc(\text{“iraqs external debt”}, D) \end{aligned}$$

where $sc(q_r, D)$ represents $\log P(q_r|D)$, which is the retrieval score of a reformulated query q_r. In our work, the sequential dependence model (SDM) is used to calculate $\log P(q_r|D)$.

8.3 Experimental Configuration

Four TREC collections, Gov2, Robust04, ClueWeb (Category B) and Wt10g are used for experiments. Robust04 is a newswire collection, while the rest are web collections. The details of the collections can be found in the Appendices. As in previous work described in this dissertation, two indexes are built for each collection, where one index is not stemmed and the other is stemmed using the Porter Stemmer (Porter 1980). No stopword removal is done during indexing. For each topic, the description part is used as the query. A short list of 35 stopwords and some frequent stop patterns (e.g., “find information”) are removed from the description query.

Table 8.2. Summary of Baselines

Name	Description
QL	query likelihood language model
SDM	sequential dependency model
KC	discovering key concept
QL+SubQL SDM+SubQL	the subset distribution approaches as described Chapter 7

The query set of each collection is split into a training set and a test set. On the training set, the parameters λ_k are learned. On the test set, the learned parameters λ_k are used to assign weight to the reformulation tree generated from each test query. Ten-fold cross validation is conducted.

Similar to previous work described in Chapter 7, the baselines include the query likelihood language model (QL), the sequential dependence model (SDM) and the key concept model (KC). Note that we do not report KC on ClueWeb, since the key concept query is not provided on ClueWeb in (Bendersky and Croft 2008). We also consider two subset query distribution models described in Chapter 7 (QL+SubQL and SDM+SubQL). They combine the original query with a distribution of subset queries. QL+SubQL uses QL for both the original query and the subset queries, while SDM+SubQL uses SDM for the original query and uses QL for the subset queries. In this chapter, QL+SubQL and SDM+SubQL are trained using the global dependency features as described in Table 7.2. The baselines used are summarized in Table 8.2.

The proposed hierarchical query distribution or reformulation tree is denoted as RTree, which uses SDM as the underlying retrieval model. Two parameters are used during the tree construction, *SubNum* and *ModNum*, where *SubNum* denotes how many subset queries are kept in the reformulation tree and *ModNum* denotes among those kept subset queries how many are further modified to generate query substitutions. In this chapter, *SubNum* takes all subset queries generated and *ModNum* is set as 10. The effect of these parameters will be explored in the following discussion.

The standard performance measures, mean average precision (MAP) and precision at 10 (P10) are used to measure retrieval performance. The details of the performance measures can be found in the Appendices. In order to improve readability, we report 100 times the actual values of these measures. The two-tailed t-test is conducted to measure significance.

8.3.1 Example

In Table 8.3, we show some examples of the generated reformulation trees. As mentioned previously, some stopwords and stop patterns are removed from the original query. Those words are kept to improve readability. Note that they are not used for retrieval and subset query generation.

Table 8.3 shows that the subset queries and the substituted queries are effectively combined within the same framework. For example, given the original query "what allegations have been made about enrons culpability in the california energy crisis", the reformulation tree first generates high quality subset queries "enrons culpability california energy crisis" and "california energy crisis" and then further modifies "california energy crisis" as two substituted queries "california gas prices" and "california electricity crisis".

8.3.2 Retrieval Performance

The first experiment is conducted to compare the retrieval performance of the proposed RTree method with the baselines. The results are shown in Table 8.4. The best performance is bolded.

Table 8.4 shows that RTree outperforms all the baseline methods. Specifically, RTree performs better than the "bag of word" representations, SDM and KC. Using the non-stemmed index, RTree significantly improves SDM and KC on all four collections. For example, on ClueWeb, RTree improves SDM by 12.2% and 16.1% with respect to MAP and P10, respectively. On Wt10g, RTree improves KC by 11.4% and

Table 8.3. Examples of the reformulation tree. The top ranked nodes are displayed. In the original query q, the stopwords and stop structures are italicized.

q: *what* allegations *have been* made about enrons culpability *in the* california energy crisis
0.194 q 0.133 enrons culpability california energy crisis 0.047 california energy crisis 0.009 california gas prices 0.008 california electricity crisis
q: *give information on* steps *to* manage control *or* protect squirrels
0.148 q 0.060 control protect squirrels 0.048 control squirrels 0.013 control of ground squirrels 0.012 control squirrel
q: *what is the* state *of* maryland *doing to* clean up *the* chesapeake bay
0.089 q 0.063 maryland chesapeake bay 0.015 md chesapeake bay 0.009 maryland chesapeake bay watershed 0.034 chesapeake bay 0.007 chesapeake bay watershed 0.006 chesapeake bay river

Table 8.4. Comparisons of retrieval performance. ⋆ denotes significantly different from the baseline.

	Gov2		Robust04		Wt10g		ClueWeb	
	MAP	P10	MAP	P10	MAP	P10	MAP	P10
Non-stemmed Index								
QL	22.46	49.13	22.40	39.12	16.44	28.97	10.97	21.63
SDM	23.98	51.01	23.30	40.04	16.76	31.65	11.53	22.76
KC	24.88	50.87	23.87	40.76	17.45	30.82	n/a	n/a
QL+SubQL	23.36	50.81	22.85	39.28	16.81	29.79	11.01	21.84
SDM+SubQL	24.82	53.36	23.65	40.32	18.25	31.13	11.54	21.94
RTree	**26.70**	**53.96**	**25.07**	**42.33**	**19.44**	**34.02**	**12.94**	**26.43**
vs. QL	18.9%⋆	9.8%⋆	11.9%⋆	8.2%⋆	18.2%⋆	17.4%⋆	18.0%⋆	22.2%⋆
vs. SDM	11.3%⋆	5.8%⋆	7.6%⋆	5.7%⋆	16.0%⋆	7.5%⋆	12.2%⋆	16.1%⋆
vs. KC	7.3%⋆	6.1%⋆	5.0%⋆	3.9%⋆	11.4%⋆	10.4%⋆	n/a	n/a
vs. QL+SubQL	14.3%⋆	6.2%⋆	9.7%⋆	7.8%⋆	15.6%⋆	14.2%⋆	17.5%⋆	21.0%⋆
vs. SDM+SubQL	7.6%⋆	1.1%	6.0%⋆	5.0%⋆	6.5%	9.3%⋆	12.1%⋆	20.5%⋆
Porter-stemmed Index								
QL	25.43	52.21	25.49	43.13	19.61	32.68	12.64	23.57
SDM	27.85	54.03	26.83	44.94	20.87	35.77	13.01	24.90
KC	27.52	53.83	25.97	41.65	21.01	34.02	n/a	n/a
QL+SubQL	26.19	53.36	25.81	43.01	20.11	32.06	12.94	25.00
SDM+SubQL	28.49	55.91	26.99	44.86	21.98	35.05	13.29	24.08
RTree	**29.85**	**56.38**	**28.00**	**45.10**	**23.80**	**37.42**	**13.69**	**25.82**
vs. QL	17.4%⋆	8.0%⋆	9.8%⋆	4.6%⋆	21.4%⋆	14.5%⋆	8.3%	9.5%
vs. SDM	7.2%⋆	4.3%⋆	4.4%⋆	0.4%	14.0%⋆	4.6%	5.2%⋆	3.7%
vs. KC	8.5%⋆	4.7%	7.8%⋆	8.3%⋆	13.3%⋆	10.0%⋆	n/a	n/a
vs. QL+SubQL	14.0%⋆	5.7%⋆	8.5%⋆	4.9%⋆	18.3%⋆	16.7%⋆	5.8%	3.3%
vs. SDM+SubQL	4.8%⋆	0.8%	3.7%⋆	0.5%	8.3%	6.8%⋆	3.0%	7.2%

10.4% with respect to MAP and P10, respectively. On the Porter-stemmed index, similar results are also observed. These results show that the "reformulation tree" representation is more effective than the "bag of words" representation on modeling verbose queries, since the former explicitly models the reformulated query, while the latter only considers words and phrases.

Furthermore, RTree also outperforms the "query distribution" representations, QL+SubQL and SDM+SubQL. Using the non-stemmed index, RTree outperforms QL+SubQL and SDM+SubQL on all four collections. Most of the improvements are significant. For example, on ClueWeb, RTree improves QL+SubQL by 17.5% and 21.0% with respect to MAP and P10, respectively. Also, RTree improves SDM+SubQL by 12.1% and 20.5% with respect to MAP and NDCG10, respectively. The results using the Porter-stemmed index are similar. These observations indicate that the "reformulation tree" representation is better than the "query distribution" representation, since the former effectively combines different reformulation operations within the same framework.

It is not surprising that RTree brings more improvements over the baselines using the non-stemmed index than using the Porter-stemmed index, since some effect of query substitutions, especially those generated using the morphologically similar words, is already provided by the Porter stemmer.

8.3.3 Further Analysis

Table 8.4 shows that RTree significantly outperforms the baseline methods. In this section, we make detailed comparisons between RTree and the baseline approaches.

First, we compare RTree with SDM and KC. Specifically, we analyze the number of queries each approach increases or decreases over QL. Fig 8.4 shows the histograms of SDM, KC and RTree based on the relative increases/decreases of MAP over QL.

Table 8.5. Comparisons with QL+SubQL and SDM+SubQL. "+", "=" and "-" denote that RTree performs better, equal or worse than QL+SubQL and SDM+SubQL with respect to MAP.

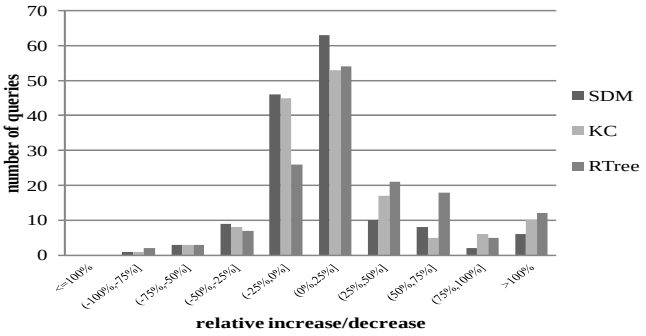
RTree	vs. QL+SubQL			vs. SDM+SubQL		
	+	=	-	+	=	-
Gov2	71.81%	0.67%	27.52%	63.09%	0.67%	36.24%
Robust04	68.27%	0.00%	31.73%	63.05%	0.00%	36.95%
Wt10g	62.89%	2.06%	35.05%	62.89%	1.03%	36.08%
ClueWeb	65.31%	2.04%	32.65%	70.41%	2.04%	27.55%

The non-stemmed index is used in Fig. 8.4. Similar results are observed using the Porter-stemmed index.

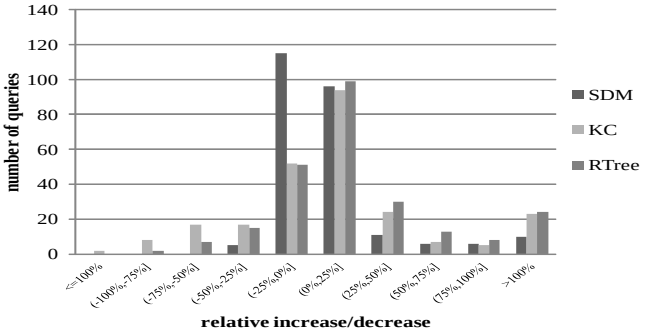
Fig. 8.4 shows that RTree improves more queries than SDM and KC. For example, on Gov2, RTree improves 110 queries out of the total 150 queries, while SDM and KC improve 89 and 91, respectively. On Robust04, RTree improves 174 queries out of the total 250 queries, while SDM and KC improve 129 and 153 queries, respectively. At the same time, RTree also hurts less queries than SDM and KC. These observations indicate that RTree is more robust than both SDM and KC.

Furthermore, we compare RTree with QL+SubQL and SDM+SubQL. QL+SubQL and SDM+SubQL only consider subset query selection, while RTree combines both subset query selection and query substitution. The comparisons between them indicate whether RTree effectively combines two query operations to improve verbose queries. Specifically, we analyze the percent of queries where RTree performs better than QL+SubQL and SDM+SubQL, respectively. The results using the non-stemmed index are reported in Table 8.5.

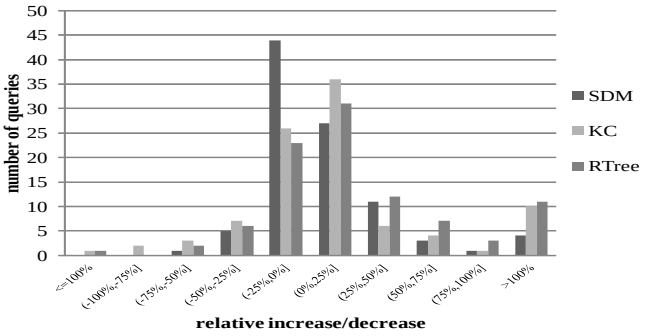
Table 8.5 shows that RTree consistently outperforms QL+SubQL and SDM+SubQL for 60%-70% queries on all four collections. These results indicate that RTree provides an effective way to combine different query operations, which significantly improves the retrieval performance of verbose queries.



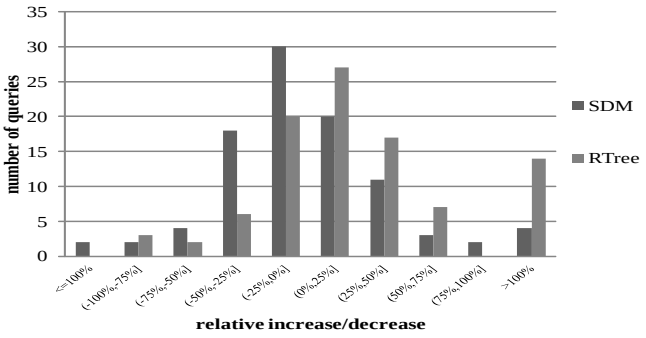
(a) Gov2



(b) Robust04



(c) Wt10g



(d) ClueWeb

Figure 8.4. Analysis of relative increases/decreases of MAP over QL.

Table 8.6. The effect of subset query selection and query substitution with respect to MAP

MAP	Gov2	Robust04	Wt10g	ClueWeb
SDM	23.98	23.30	16.76	11.53
KC	24.88	23.87	17.45	n/a
QL+SubQL	23.36	22.85	16.81	11.01
SDM+SubQL	24.82	23.65	18.25	11.54
RTree-Subset	25.80	24.76	18.11	11.73
RTree	26.70	25.07	19.44	12.94

8.3.4 Subset Selection vs. Query Substitution

RTree combines subset query selection and query substitution together using a two-level reformulation tree. Previous experiments have demonstrated the general effect of this approach. In this part, we split the effect of subset query selection and query substitution. Specifically, we propose a one-level reformulation tree, which only consists of subset queries. This one-level reformulation tree is denoted as RTree-Subset. The comparisons between RTree-Subset and other approaches using the non-stemmed index are shown in Table 8.6.

In Table 8.6, RTree-Subset outperforms the baseline methods, which indicates the effect of subset queries in the reformulation tree. When query substitutions are introduced, RTree further improves RTree-Subset. Thus, both subset selection and query substitution account for the performance of RTree. RTree-Subset also performs better than other subset query selection methods such as QL+SubQL and SDM+SubQL.

8.3.5 Parameter Analysis

As described in Section 4.1, there are two parameters used during the process of constructing the reformulation tree, $SubNum$ and $ModNum$. $SubNum$ denotes the number of subset queries used in the reformulation tree and $ModNum$ denotes the number of subset queries that are modified to generate query substitutions. In this subsection, we explore the effect of these two parameters. The Porter-stemmed index

Table 8.7. The effect of the parameter $ModNum$ with respect to MAP

$ModNum$	Gov2	Robust04	Wt10g	ClueWeb
3	29.52	27.08	22.61	13.77
5	29.63	27.11	22.72	13.75
10	29.66	27.16	22.75	13.71

is used. Fig. 8.5 shows the effect of the parameter $SubNum$, where $SubNum$ takes the values 5, 10, 20, 30 and "all", where $ModNum$ is fixed as 10. Here, "all" indicates using all subset queries generated.

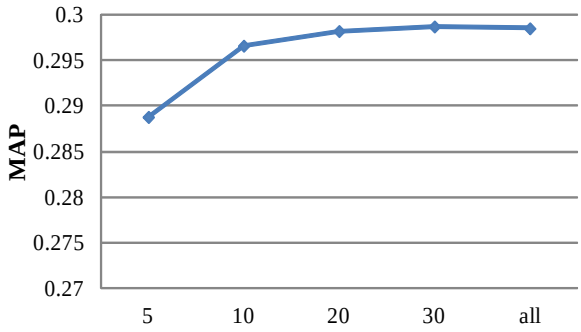
Fig. 8.5 shows that the best number of subset queries used in the reformulation tree is inconsistent. On Gov2, the performance becomes stable after using the top 20 subset queries. On Robust04 and Wt10g, the performance keeps increasing when more subset queries are considered. On ClueWeb, the performance drops when more than the top 20 queries are used. One possible explanation for these observations is provided. Robust04 and Wt10g are relatively small collections, thus using more subset queries is likely to retrieve more relevant documents. However, when the size of the collection becomes very large such as Gov2, using more subset queries does not help much for retrieving more relevant documents. If the collection is not only big but also contains much noise such as ClueWeb, using more subset queries even hurts the performance.

Table 8.7 displays the retrieval performance when $ModNum$ takes three different values, i.e. 3, 5 and 10, where $SubNum$ is set as 10.

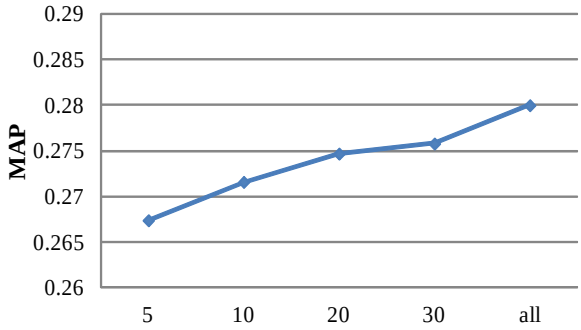
Table 8.7 shows that there is little performance change when $ModNum$ is bigger than 3, which indicates that modifying the top three subset queries is enough to achieve similar performance to RTree.

8.4 Summary

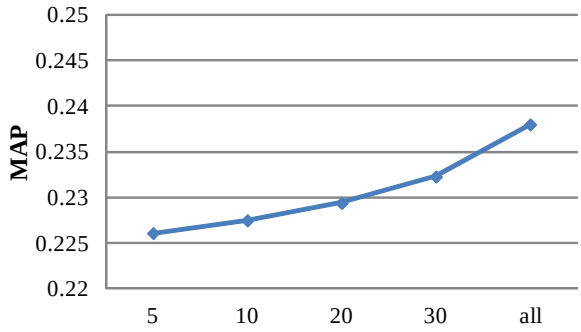
Complex queries pose a new challenge to search systems. In order to combine different query operations and model the relationships between the reformulated queries,



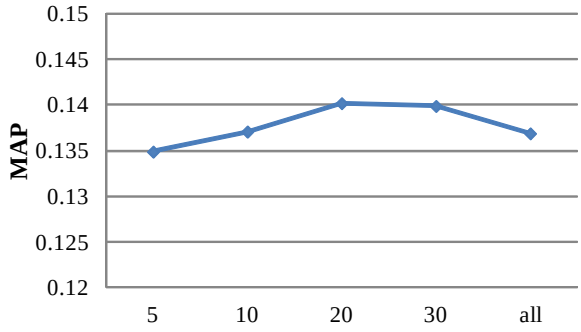
(a) Gov2



(b) Robust04



(c) Wt10g



(d) ClueWeb

Figure 8.5. The effect of the parameter *SubNum*. x-axis denotes *SubNum* and y-axis denotes MAP.

a new query representation is proposed, where the original query is transformed into a reformulation tree. A specific implementation is described for verbose queries, which combines subset query selection and query substitution within a principled framework. In the future, this query representation will be applied to other search tasks involving complex queries such as the cross-lingual retrieval and diversifying the search results.

CHAPTER 9

CONCLUSION AND FUTURE WORK

Query reformulation modifies the original query posed by a user to provide a better representation of the underlying information need for a search system. In this dissertation, we propose a novel reformulation framework that transforms the original query into a distribution of reformulated queries, where each reformulated query is associated with a probability indicating its importance for retrieval. The query distribution model considers a reformulated query as the basic unit, thus explicitly modeling how query concepts are used together to form a realistic or actual query. Since a reformulated query is the output of applying single or multiple query operations, different reformulation operations such as query segmentation and query substitution are combined within the same framework. Also, this model provides a joint view of the reformulation model and the retrieval model, where the probability assigned to the reformulated query is estimated by directly optimizing the retrieval performance.

In this chapter, we first summarize the content of this dissertation and then restate our contributions. We also discuss future directions.

9.1 Summary

The summary of this dissertation is as follows.

- **Chapter 3 Framework of Query Distributions.** We introduce the general framework of the query distribution model (QDist), which consists of three major components, reformulated query generation, probability estimation and

retrieval. Then, we compare the query distribution model with two types of reformulation models, i.e, Concept Distribution (CDist) and Single Reformulated Query (SRQ). Compared with CDist, QDist explicitly models how query concepts are used to form realistic or actual queries and thus capture the query-level dependencies of concepts. Compared with SRQ, QDist considers alternative reformulated queries.

- **Chapter 4 Reformulated Query Generation.** We describe four approaches of generating reformulated queries according to different types of queries and applications. Specifically, for short keyword queries, we develop the passage analysis technique that uses the passages containing all query words or most query words to generate query segmentation and substitution. For verbose queries, we select a subset of query words to form the subset queries. Several strategies are used to filter the number of subset queries. For natural language questions, we search a large scale Q&A archive to find semantically related questions and use them as reformulated queries. A translation-based language model is proposed to search the Q&A archive. In patent retrieval, we transform the patent application into several Indri queries.

- **Chapter 5 Probability Estimation.** We develop two probability estimation approaches that estimate the probability of each reformulated query by directly optimizing the retrieval performance. Specifically, the first approach optimizes the retrieval performance of using a single reformulated query, while the second approach optimizes the performance of using a set of reformulated queries.

- **Chapter 6 Application I: Query Distribution for Short Queries.** We present an implementation of the query distribution model for short queries. This implementation uses the passage analysis technique to generate reformulated queries and optimizes the performance of using a set of reformulated query.

The query likelihood language model is used as the underlying retrieval model. Experiments conducted on three TREC collections show the advantages of the query distribution model over state-of-the-art techniques.

- **Chapter 7 Application II: Query Distribution for Verbose Queries.** We present an implementation of the query distribution model for verbose queries. This implementation uses the subset selection to generate subset queries and optimizes the performance of using a single reformulated query. The query likelihood language model and the sequential dependence model are used as the underlying retrieval models. Experiments conducted on three TREC collections show that the query distribution model outperforms other query processing techniques developed for verbose queries.

- **Chapter 8 Hierarchical Query Distribution.** We propose the hierarchial query distribution model to process complex queries and capture the relationships between the reformulated queries. A two-level reformulation tree is implemented for verbose queries, which combines the subset selection and the query substitution. A stage-based probability estimation approach is developed. Experiments on four TREC collections show that the hierarchical query distribution model performs better than state-of-the-art techniques and also outperforms the standard query distribution model.

9.2 Contributions

Our major contributions through this dissertation are restated as follows:

1. **Novel Reformulation Framework.** We propose a novel framework for query reformulation, where the original query is transformed into a distribution of reformulated queries and the retrieval model is considered as an integrated part of this framework.

2. **Novel Probability Estimation Approaches.** We develop two approaches to estimate the probability of each reformulated query by directly optimizing the retrieval performance. These probability estimation approaches consider the reformulation model and the retrieval model jointly.

3. **Hierarchical Query Distribution for Complex Queries.** We propose a hierarchical query distribution that extends the standard query distribution model to capture the dependencies between the reformulated queries. This model is useful for dealing with complex queries that require a series of reformulation operations.

4. **Effective Reformulated Query Generation Approaches.** We design several effective approaches to generate the reformulated queries that are semantically related to the original query. These approaches work well for short queries, verbose queries, natural language questions and patent applications.

5. **State of the art retrieval performance.** We test the query distribution model on different applications and achieve consistent and significant performance improvement over state-of-the-art techniques.

9.3 Future Work

In this dissertation, we build the foundation of the query distribution model for query reformulation. Although we have solved several critical issues, some challenges still exist, which will be explored in the future work.

- **Utilizing Query Logs.** In this work, we use different approaches such as passage analysis and subset selection to generate the reformulated queries as a simulation of realistic or actual user queries. Although these techniques are effective on TREC collections, we expect to show the full potential of the query distribution model when large scale query logs are available. On one hand,

realistic user queries that are semantically related to the original query can be extracted using the clickthrough data and the session information. On the other hand, many valuable features characterizing the whole query are available. These features include the query log frequency and the number of clicks after issuing a query. Using these high quality reformulated queries and features, the performance of the query distribution model can be further improved.

- **Efficient implementation.** The efficiency issue is critical for implementing the query distribution model in a practical system. The online cost of the query distribution model comes from three aspects, i.e., generating reformulated queries, extracting query features and running multiple queries. The first two aspects can be efficiently implemented when large scale query logs are available. We can limit the reformulated queries to those appearing in query logs. In this way, instead of generating queries, we simply search the query logs, which can be efficiently implemented using the index. Also, all query features can be precomputed, which speeds up the query feature extraction. For the retrieval aspect, instead of running multiple reformulated queries, we reuse the retrieval scores of the words and phrases shared by these queries.

- **Modeling the dependencies between the reformulated queries.** There are different types of relationships between the reformulated queries. In Chapter 8, we model the sequence of generating reformulated queries as a hierarchical query distribution. Another type of relationship between the reformulated queries is their similarities. In the query distribution, instead of assigning high probabilities to similar reformulated queries, we may want to diversify the probabilities to the reformulated queries with different properties. In particular, how to incorporate query similarities into the objective function of probability estimation is an interesting research issue. Furthermore, it would also be interesting

to investigate the connection between diversifying the reformulated queries and diversifying the search results.

- **Jointly optimizing reformulation and retrieval.** In Chapter 5, we study optimizing the reformulation model for a given retrieval model. On the other hand, many learning-to-rank techniques focus on optimizing the retrieval model using a given query. It is natural to consider jointly optimizing both the reformulation model and the retrieval model. In other words, the parameters of the reformulation model and the parameters of the retrieval model should be learned in a unified framework. How to design the joint learning strategy is a very important research issue.

APPENDIX A

BACKGROUND

A.1 Retrieval Models

The query likelihood language model (Ponte and Croft 1998; Zhai and Lafferty 2001b) is widely used in information retrieval, where the ranking score of a document is calculated as the probability of generating the query from this document. All retrieval models discussed in this dissertation use the language model as the basis. Given a query q, a document D in the collection C is ranked by $\log P(q|D)$, which is calculated as follows:

$$\log P(q|D) = \sum_{w \in q} \log P(w|D) = \sum_{w \in q} \log[\lambda P_{ml}(w|D) + (1-\lambda) P_{ml}(w|C)] \qquad \text{(A.1)}$$

In Eq. A.1, $P(w|D)$ is the probability of generating a word w from a document D. It is a mixture of the maximum likelihood estimation $P_{ml}(w|D)$ and the background model $P_{ml}(w|C)$. $P_{ml}(w|D)$ and $P_{ml}(w|C)$ are calculated as follows.

$$P_{ml}(w|D) = \frac{\#(w, D)}{|D|} \qquad \text{(A.2)}$$

$$P_{ml}(w|C) = \frac{\#(w, C)}{|C|} \qquad \text{(A.3)}$$

where $\#(w, D)$ is the frequency of w in a document D and $\#(w, C)$ is the frequency of w in the whole collection C. $|D|$ is the number of words in D. Similarly, $|C|$ is the total number of words in the whole collection.

In Eq. A.1, λ is the smoothing parameter. Several smoothing methods are studied for the language model (Zhai and Lafferty 2001b). The Jelinek-Mercer method assigns a fixed value to λ. The Dirichlet method set λ as $\frac{|D|}{|D|+\mu}$, where μ is a Dirichlet parameter. This method assigns high λ values to long documents. The two-stage method combines the Jelinek-Mercer method and the Dirichlet method.

The sequential dependence model (SDM) (Metzler and Croft 2005) provides a unified framework to model the phrase and the term proximity. Using SDM, the retrieval score of a document D, i.e., $\log P(q|D)$, is calculated as follows:

$$\begin{aligned} \log P(q|D) &= \lambda_T \sum_{t \in T(q)} \log P(t|D) + \lambda_O \sum_{o \in O(q)} \log P(o|D) \\ &\quad + \lambda_U \sum_{u \in U(q)} \log P(u|D) \end{aligned} \tag{A.4}$$

where $T(q)$ denotes a set of query words of q, $O(q)$ denotes a set of ordered bigrams extracted from q and $U(q)$ denotes a set of unordered bigrams extracted from q. λ_T, λ_O and λ_U are parameters controlling the weights of different parts and are usually set as 0.85, 0.1 and 0.05 (Metzler and Croft 2005). $P(t|D)$, $P(o|D)$ and $P(u|D)$ are calculated using the language modeling approach.

A.2 Indri Query Language

Indri[1] is a search engine that combines the inference network with the language model (Metzler and Croft 2004). The Indri query language supports many advanced features such as phrase, synonym, term proximity and weighted expressions.

The query likelihood language model and the sequence dependence model can be easily implemented using the Indri query language. For example, given the original

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

Figure A.1. Indri queries implementing the query likelihood language model (QL) and the sequential dependence model (SDM) .

```
q : iraqs foreign debt

QL: #combine(iraqs foreign debt)

SDM: #weight(
        0.85 #combine(iraqs foreign debt)
        0.10 #combine(#1(iraqs foreign)  #1(foreign debt))
        0.05 #combine(#uw8(iraqs foreign)  #uw8(foreign debt))
      )
```

query "iraqs foreign debt", the Indri queries that implement the query likelihood language model and the sequential dependence model are displayed in Fig. A.1.

In Fig. A.1, "#combine" is an unweighted combinator that equally combines the scores of several expressions. An expression could be a word, a phrase, term proximity or a combination of them. "#combine(iraqs foreign debt)" implements Eq. A.1, where the score of each query word, i.e., $\log P(w|D)$, is equally combined. "#1" indicates a phrase or an ordered bigram, while "#uwN" indicates an unordered bigram in a word window with the size N. In Fig. A.1, "#combine(#1(iraqs foreign) #1(foreign debt))" and "#combine(#uw8(iraqs foreign) #uw8(foreign debt))" implement the ordered bigram part, i.e., $\sum_{o \in O(q)} \log P(o|D)$, and the unordered bigram part, i.e., $\sum_{u \in U(q)} \log P(u|D)$, of Eq. A.4, respectively. "#weight" is a weighted combinator that combines several expressions using their corresponding weights. In Fig. A.1, SDM combines three expressions using the weight 0.85, 0.10 and 0.05, respectively.

A.3 Collections

Four TREC collections are used in this dissertation. The properties of these collections are quite different. The details are summarized in Table A.1. Robust04 is a small newswire collection, thus it is a homogeneous collection with the controlled vocabulary. Wt10g is a relatively small web collection. Compared with Robust04,

Table A.1. TREC collections used in experiments

Name	Description	Docs	Topics
Robust04	newswire from different sources	528,155	301-450,601-700
Wt10g	small web collection	1,692,096	451-550
Gov2	crawl of .gov domain in 2004	25,205,179	701-850
ClueWeb (Category B)	crawl of web in 2009	50,220,423	1-100

Figure A.2. An example of TREC topic

Number: 705

Title: iraqs foreign debt reduction

Description: Identify any efforts, proposed or undertaken, by world governments to seek reduction of Iraq's foreign debt.

Narrative: Documents noting this subject as a topic for discussion (e.g. at U.N. and G7) are relevant. Money pledged for reconstruction is irrelevant.

it is a heterogenous collection and contains much noise. Gov2 is a large scale web collection. Since only the .gov domain is crawled, it contains less noise. Clueweb is the largest collection currently available. It contains around 1 billion web pages in ten languages and these web pages were crawled in 2009. The category B consists of the first 50 million English pages with relatively high quality. Also, there is much noise in Clueweb.

The topics used in each collection represent users' information needs. Fig. A.2 provides an example of TREC topic, which consists of several fields, i.e., Number, Title, Description and Narrative. Number indicates the identity of a TREC topic. Title and Description both describe the information need. Title uses a small number of keywords , while Description uses the natural language. In this disseration, Title and Description are used as short keyword queries and verbose queries, respectively. The Narrative field provides the instructions for human annotators, who are asked to decide the relevance of a document.

Given a topic, the relevance judgments are also provided in TREC collections. Specifically, a set of documents are manually judged for relevance. Besides the binary judgments, the graded judgments are also provided, especially for web collections. Due to the size of a collection, it is impossible for human annotators to judge every document for a given topic. Instead, the pooling strategy is adopted to collect a reasonable amount of documents for judging and ensure that these judgments are reusable for future systems.

A.4 Evaluation Metrics

Given the relevant judgments of each topic, several evaluation metrics can be calculated to measure the performance of a system. Two types of metrics are used in this dissertation, mean average precision (MAP) and precision at k (P@k).

The average precision and the precision at k of a given query are calculated as follows.

$$AP = \frac{1}{R} \sum_{i:\ rel(i)=1} \frac{R(1,i)}{i} \tag{A.5}$$

$$P@k = \frac{1}{k}\ R(1,k) \tag{A.6}$$

$$R(1,i) = \sum_{j=1}^{i} rel(j)$$

$rel(i)$ denotes the relevance of a document at the position i of a ranked list. If this document is relevant, $rel(i) = 1$, otherwise $rel(i) = 0$. $R(1,i)$ denotes the number of relevant documents till the position i. R is the total number of relevant documents. The average precision measures the quality of the entire ranked list, while the precision at k measures the quality of the top ranked documents.

In order to aggregate the above metrics over a set of queries, we calculate the arithmetic mean as follows.

$$MAP = \frac{1}{N}\sum_{i=1}^{N} AP(q_i) \tag{A.7}$$

$$Pk = \frac{1}{N}\sum_{i=1}^{N} P@k(q_i) \tag{A.8}$$

N is the total number of queries. MAP denotes the mean average precision. Pk denotes the average precision at k.

APPENDIX B

PROOFS

Claim 1

$$P(c|q) = \sum_{q_r \in \{q_r | c \in q_r\}} P(q_r|q), \text{ given that } P(q_r|q) = \delta(|q_r|) \prod_{i=1}^{|q_r|} P(c_i|q)$$

Proof.

$$\begin{aligned}
& P(c|q) \\
= \; & P(c|q) \sum_{l=1}^{\infty} \delta(l) && \text{(B.1)} \\
= \; & P(c|q) \sum_{l=1}^{\infty} \delta(l) \prod_{i=1}^{l-1} \sum_{c_i} P(c_i|q) && \text{(B.2)} \\
= \; & P(c|q) \sum_{l=1}^{\infty} \delta(l) \sum_{q'_r \in \{|q'_r|=l-1\}} \prod_{c_i \in q'_r} P(c_i|q) && \text{(B.3)} \\
= \; & \sum_{l=1}^{\infty} \sum_{q_r \in \{(q'_r, c)\}} \delta(l) \prod_{c_i \in q'_r} P(c_i|q) \cdot P(c|q) && \text{(B.4)} \\
= \; & \sum_{l=1}^{\infty} \sum_{q_r \in \{c \in q_r, |q_r|=l\}} P(q_r|q) && \text{(B.5)} \\
= \; & \sum_{q_r \in \{q_r | c \in q_r\}} P(q_r|q) && \text{(B.6)}
\end{aligned}$$

Eq. B.1 is obtained, since δ is a distribution over the possible length of a query, i.e., $\sum_{l=1}^{\infty} \delta(l) = 1$. Eq. B.2 is obtained, since $P(c_i|q)$ is a multinomial distribution, i.e., $\sum_{c_i} P(c_i|q) = 1$. Then, we reorder the sequence of production and summation to obtain Eq. B.3, where q'_r represents any query with the length of $l-1$. In Eq. B.4,

we concatenate q'_r and c to get q_r, where the length of q_r is l. Eq. B.5 is obtained using the condition provided, i.e., $P(q_r|q) = \delta(|q_r|) \prod_{i=1}^{|q_r|} P(c_i|q)$. Finally, in Eq. B.6, we sum over all possible lengths l to prove the claim. □

BIBLIOGRAPHY

Allan, J., W. B. Croft, D. Fisher, M. Connell, F. Feng, and X. Li, 2000 INQUERY and TREC-9. In *Proceedings of TREC-9*, pp. 551–562.

Bendersky, M. and W. B. Croft, 2008 Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, pp. 491–498.

Bendersky, M., D. Metzler, and W. Croft, 2011 Parameterized concept weighting in verbose queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in information retrieval*, Beijing, China, pp. 605–614.

Bendersky, M., D. Metzler, and W. B. Croft, 2010 Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining*, New York City, NY.

Bendersky, M., D. A. Smith, and W. B. Croft, 2009 Two-stage query segmentation for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, pp. 810–811.

Bergsma, S. and Q. I. Wang, 2007 Learning noun phrase query segmentation. In *Proceedings of the 2007 joint conference on Empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, Prague, Czech Republic, pp. 819–826.

Billerbeck, B., F. Scholer, H. E. Williams, and J. Zobel, 2003 Query expansion using associated queries. In *Proceedings of the 12th ACM international conference on Information and knowledge management*, New Orleans, LA, pp. 2–9.

Boldi, P., F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, 2008 The query-flow graph: model and applications. In *Proceedings of the 17th ACM international conference on Information and knowledge management*, Napa Valley, CA, pp. 609–618.

Boldi, P., F. Bonchi, C. Castillo, and S. Vigna, 2009 From "Dango" to "Japanese Cakes": query reformulation models and patterns. In *Web Intelligence and Intelligent Agent Technologies*, Volume 1, pp. 183–190. IEEE.

Brown, P. F., V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, 1993 The mathematics of statistical machine translation: paramter estimation. Computational Linguistics *19*(2)**:** 263–311.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, 2005 Learning to rank using gradient decent. In *Proceedings of the 22nd international conference on Machine learning*, Bonn, Germany, pp. 89–96.

Byrd, R. H., J. Nocedal, and R. B. Schnabel, 1994 Rrepresentations of quasi-Newton Matrices and their use in limited memory methods. Mathematical Programming *63*(2)**:** 129–156.

Callan, J. and W. Croft, 1993 An evaluation of query processing strategies using the TIPSTER collection. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, Pittsburgh, PA, pp. 347–355.

Cao, G., J. Y. Nie, J. Gao, and S. Robertson, 2008 Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, pp. 243–250.

Cao, H., D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, 2008 Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, Las Vegas, NE, pp. 875–883.

Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, 2007 Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136.

Chien, S. and N. Immorlica, 2005 Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web*, Chiba, Japan, pp. 2–11.

Chirita, P.-A., C. S. Firan, and W. Nejdl, 2007 Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, the Netherlands, pp. 7–14.

Collins-Thompson, K., 2008 Robust model estimation methods for information retrieval. Ph. D. thesis, Carnegie Mellon University.

Collins-Thompson, K. and J. Callan, 2007 Estimatin and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdan, Netherland, pp. 303–310.

Cronen-Townsend, S., Y. Zhou, and W. B. Croft, 2002 Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere, Finland, pp. 299–306.

Crouch, C. J. and B. Yang, 1992 Experiments in automatic statistical thesaurus construction. In *Proceedings of the 15th annual international ACM SI-*

GIR conference on Research and development in information retrieval, Copenhagen, Denmark, pp. 77–88.

Cucerzan, S. and E. Brill, 2006 Extracting semantically related queries by exploiting user session information. In *research.microsoft.com/en-us/people/silviu/np-www06.pdf*.

Cui, H., J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, 2002 Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web*, Honolulu, Hawaii, pp. 325–332.

Dang, V. and W. B. Croft, 2010 Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*, New York, NY, pp. 41–50.

De Marneffe, M. and C. Manning, 2008 Stanford typed dependencies manual. URL http://nlp. stanford. edu/software/dependencies_manual. pdf.

Freund, Y., R. D. Iyer, R. E. Schapire, and Y. Singer, 2003 An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4:** 933–969.

Guo, J., G. Xu, H. Li, and X. Cheng, 2008 A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, pp. 379–386.

He, B. and I. Ounis, 2004 Inferring query performance using pre-retrieval predictors. In *String Processing and Information Retrieval*, pp. 43–54.

Herbrich, R., T. Graepel, and K. Obermayer, 2000 *Large margin rank boundaries for ordinal regression*. MIT Press: Cambridge, MA.

Huang, J., J. Gao, J. Miao, X. Li, K. Wang, F. Behr, and C. L. Giles, 2010 Exploring web scale language models for search query processing. In *Proceedings of the 19th international conference on World wide web*, Raleigh, NC, pp. 451–460.

Huston, S. and W. B. Croft, 2010 Evaluating verbose query processing techniques. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, Geneva, Switzerland, pp. 291–298.

Jansen, B., D. Booth, and A. Spink, 2009 Patterns of query reformulation during web searching. Journal of the American Society for Information Science and Technology *60*(7)**:** 1358–1371.

Jeon, J., W. B. Croft, and J. H. Lee, 2005 Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 84–90.

Jing, Y. and W. B. Croft, 1994 An association thesaurus for information retrieval. In *Proceedings of RIAO*, Volume 94, pp. 146–160.

Joachims, T., 2002 Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Alberta, Canada, pp. 133–142.

Jones, R. and D. C. Fain, 2003 Query word deletion prediction. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, Toronto, Canada, pp. 435–436.

Jones, R., B. Rey, O. Madani, and W. Greiner, 2006 Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, Ediburgh, Scotland, pp. 387–396.

Jurafsky, D., J. Martin, A. Kehler, K. Vander Linden, and N. Ward, 2000 *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*, Volume 2. Prentice Hall New Jersey.

Klein, D. and C. D. Manning, 2003 Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan, pp. 423–430.

Krovetz, R., 1993 Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, Pittsburgh, PA, pp. 191–202.

Kukich, K., 1992 Techniques for automatically correcting words in text. ACM Computing Surveys (CSUR) *24*(4)**:** 377–439.

Kumaran, G. and J. Allan, 2007 A case for shorter queries, and helping users creat them. In *Proceedings of NAACL-HLT*, Rochester, NY, pp. 220–227.

Kumaran, G. and V. R. Carvalho, 2009 Reducing long queries using query quality predictors. In *Proceeding of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, pp. 564–571.

Lang, H., D. Metzler, B. Wang, and J. Li, 2010 Improved latent concept expansion using hierarchical markov random fields. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 249–258. ACM.

Lavrenko, V. and W. B. Croft, 2001 Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, LA, pp. 120–127.

Lease, M., 2009 An improved Markov random field model for supporting verbose queries. In *Proceedings of the 32nd annual international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, pp. 476–483.

Lease, M., J. Allan, and W. B. Croft, 2009 Regression rank: learning to meet the opportunity of descriptive queries. Advances in Information Retrieval**:** 90–101.

Lee, K. S., W. B. Croft, and J. Allan, 2008 A cluster-based resampling method for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, pp. 235–242.

Liu, X. and W. B. Croft, 2002 Passage retrieval based on language models. In *Proceedings of the 11th ACM international conference on Information and knowledge management*, McLean, VA, pp. 375–382.

Lo, R., B. He, and I. Ounis, 2005 Automatically building a stopword list for an information retrieval system. Proc. of DIR **5**.

Lv, Y. and C. Zhai, 2010 Positional relevance model for pseudo-relevance feedback. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, Geneva, Switzerland, pp. 579–586.

Mei, Q., K. Klinkner, R. Kumar, and A. Tomkins, 2009 An analysis framework for search sequences. In *Proceeding of the 18th ACM international conference on Information and knowledge management*, Hong Kong, China, pp. 1991–1994.

Metzler, D. and W. B. Croft, 2004 Combining the language model and inference network approaches to retrieval. Information Processing and Management *40*(5)**:** 735–750.

Metzler, D. and W. B. Croft, 2005 A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, pp. 472–479.

Metzler, D. and W. B. Croft, 2007 Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, the Netherlands, pp. 311–318.

Peng, F., N. Ahmed, X. Li, and Y. Lu, 2007 Context sensitive stemming for web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, the Netherlands, pp. 639–646.

Ponte, J. M. and W. B. Croft, 1998 A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, Melbourne, Australia, pp. 275–281.

Porter, M. F., 1980 An algorithm for suffix stripping. Program *14*(3)**:** 130–137.

Qin, T., T.-Y. Liu, X.-D. Zhang, D.-S. Wang, W.-Y. Xiong, and H. Li, 2008 Learning to rank relational objects and its application to web search. In *Proceeding of the 17th international conference on World Wide Web*, Beijing, China, pp. 407–416.

Qiu, Y. and H. P. Frei, 1993 Concept based query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, Pittsburgh, PA, pp. 160–169.

Robertson, S., 2005 On GMAP and other transformations. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, Arlington, VA, pp. 78–83.

Rocchio, J. J., 1971 *The SMART Retrieval System - Experiments in Automatic Document Processing*, Chapter Relevance feedback in information retrieval. Englewood Cliffs, NJ: Prentice Hall.

Sheldon, D., M. Shokouhi, M. Szummer, and N. Craswell, 2011 Lambdamerge: merging the results of query reformulations. In *Proceedings of the fourth ACM international conference on Web search and data mining*, Hong Kong, China, pp. 795–804.

Sun, X., J. Gao, D. Micol, and C. Quirk, 2010 Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, pp. 266–274.

Tan, B. and F. Peng, 2008 Unsupervised query segmentation using generative language models and wikipedia. In *Proceeding of the 17th international conference on World Wide Web*, Beijing,China, pp. 347–356.

van Rijsbergen, C., 1979 *Information Retrieval* (2nd ed.). London: Butterworths.

Vlachos, M., C. Meek, Z. Vagena, and D. Gunopulos, 2004 Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the ACM SIGMOD international conference on Management of data*, Paris, France, pp. 131–142.

Wang, L., J. Lin, and D. Metzler, 2010 Learning to efficiently rank. In *Proceedings of the 33th international ACM SIGIR conference on Research and development in Information*, Geneva, Switzerland, pp. 138–145.

Wang, X. and C. Zhai, 2008 Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM conference on Information and knowledge management*, Napa Valley, CA, pp. 479–488.

Xu, J. and W. Croft, 1998 Corpus-based stemming using cooccurrence of word variants. ACM Transactions on Information Systems (TOIS) *16*(1)**:** 61–81.

Xu, J. and W. B. Croft, 2000 Improving the effectiveness of information retrieval with local context analysis. ACM Transactions on Information Systems *18*(1)**:** 79–112.

Xu, J. and H. Li, 2007 AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, the Netherlands, pp. 391–398.

Xu, Y., G. J. Jones, and B. Wang, 2009 Query dependent pseudo-relevance feedback based on wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, pp. 59–66.

Xue, X. and W. B. Croft, 2009 Automatic query generation for patent search. In *Proceeding of the 18th ACM conference on Information and knowledge management*, Hong Kong, China, pp. 2037–2040.

Xue, X. and W. B. Croft, 2010 Representing queries as distributions. In *SIGIR10 Workshop on Query Representation and Understanding*, Geneva, Switzerland, pp. 9–12.

Xue, X. and W. B. Croft, 2011 Modeling subset distributions for verbose queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in information retrieval*, Beijing, China, pp. 1133–1134.

Xue, X. and W. B. Croft, 2012 Generating reformulation trees for complex queries. In *Proceeding of the 35th international ACM SIGIR conference on Research and development in information retrieval*, Portland, Oregon, pp. to appear.

Xue, X., W. B. Croft, and D. A. Smith, 2010 Modeling reformulation using passage analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, Toronto, ON, Canada, pp. 1497–1500.

Xue, X., S. Huston, and W. B. Croft, 2010 Improving verbose queries using subset distribution. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, Toronto, ON, Canada, pp. 1059–1068.

Xue, X., J. Jeon, and W. B. Croft, 2008 Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 475–482.

Zhai, C. and J. Lafferty, 2001a Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth ACM international conference on Information and knowledge management*, Atlanta, GA, pp. 403–410.

Zhai, C. and J. Lafferty, 2001b A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, LA, pp. 334–342.