Robust model estimation methods for information retrieval

Kevyn B. Collins-Thompson

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Thesis Committee:
Jamie Callan, *Chair*
William Cohen
Susan Dumais (Microsoft Research)
John Lafferty

December 4, 2008

Abstract

Information retrieval algorithms attempt to match a user's description of their information need with relevant information in a collection of documents or other data. Applications include Web search engines, filtering and recommendation systems, computer-assisted language tutors, and many others. A key challenge of retrieval algorithms is to perform effective matching when many factors, such as the user's true information need, may be highly uncertain and can only be partially observed via a small number of keywords. This dissertation develops broadly applicable algorithms for measuring and exploiting such uncertainty in retrieval algorithms to make them more effective and reliable. Our contributions include new theoretical models, statistical methods, evaluation techniques, and retrieval algorithms.

As an application, we focus on a long-studied approach to improving retrieval matching that adds related terms to a query – a process known as *query expansion*. Query expansion works well on average, but even state-of-the-art methods are still highly unreliable and can greatly hurt results for individual queries. We show how sensitivity information for an expansion algorithm can be obtained and used to improve its reliability without reducing overall effectiveness.

Our approach proceeds in two steps. First, treating the base expansion method as a 'black box', we gather information about how the algorithm's output – a set of expansion terms – changes with perturbations of the initial query and top-ranked documents. This step also results in a set of plausible expansion model candidates. We then introduce a novel risk framework based on convex optimization that prunes and combines these candidates to produce a much more reliable version of the original baseline expansion algorithm. Highlights of our results include:

- A new algorithmic framework for estimating more precise query and document models, based on treating queries and document sets as random variables instead of single

observations.

- The first significant application and analysis of convex optimization methods to query expansion problems in information retrieval.
- A new family of statistical similarity measures we call *perturbation kernels* that are efficient to compute and give context-sensitive word clustering.
- The introduction of risk-reward analysis to information retrieval, including tradeoff curves, analysis, and risk measures.
- A new general form of query difficulty measure that reflects clustering in the collection as well as the relation between a query and the collection.

Acknowledgements

Many people have helped me come to this place in my education and career. I am sincerely grateful to all the faculty, staff, and students at Carnegie Mellon who helped me in ways large and small. Of these, my greatest debt is to my advisor, Jamie Callan. Because of Jamie's confidence in me, I was able to explore research directions with great freedom, while getting guidance from him at critical times with new suggestions, insights, and honest feedback. Jamie was also truly generous and understanding of my family needs throughout graduate school. I am fortunate to have had Jamie as my advisor and have learned much from him.

I'm also grateful to my thesis committee, William Cohen, Susan Dumais, and John Lafferty. John has been very supportive throughout my time at CMU, and in addition to his valuable feedback on my thesis, was also the one who opened my eyes to the beauty of statistical language modeling and convex optimization. William's keen insights, feedback, and encouragement were much appreciated, and conversations with him helped me formulate the graph labeling model. Sue has been a valued mentor from the start of my research career and it is hard to say enough about her incredible energy, analytical abilities, and knowledge of the field of information retrieval.

The ideas in this thesis were shaped by many people, especially Guy Lebanon, Fernando Diaz, Oren Kurland, Paul Bennett, Paul Ogilvie, Luo Si, Trevor Strohman, Chengxiang Zhai and Don Metzler. Working with Chris Buckley and others at the Reliable Information Access workshop in the summer of 2003 was another important influence.

I am particularly thankful for the friendship of Paul Bennett through both real and academic marathons, and Paul Ogilvie, to whom I owe many good times (and crash space) starting with my very first trip to Pittsburgh. I also thank Jaime Arguello, Vitor Carvalho, Jon Elsas, Michael Heilman, Kornel Laskowski, Vasco Pedro, John Kominek, my officemate Li Fan, Yi Zhang, Kathrin Probst, and Jimi Shanahan for making life that much more

fun and interesting. I'm grateful to Camille Goudeseune for many years of moral support. The memory of my late friend Simon Gibson and his amazing zest for life has always stayed with me and inspired me. When things got tough, Simon often reminded me to step back and look at life 'as big as the moon'.

My remote working in the last year was made much more fun and social thanks to Jacob, Susan and the rest of the Nomads at Office Nomads, and to the friendly baristas at Caffe Ladro (Upper Queen Anne) in Seattle.

My parents David and Kate have been eternally loving and supportive. They were the ones who instilled in me a love of learning and science, and the self-confidence to leap into the unknown.

Finally, none of this work would have been possible without the unwavering love and support of my wife Alice, who spent many long days and sleepless nights, sometimes alone, caring for the kids while I was working on the research for this thesis. I dedicate this thesis to her, and to Henry and Karenna, who are the most important and inspiring outcomes from our time in Pittsburgh.

For Alice, Karenna and Henry

Contents

List of Figures

List of Tables

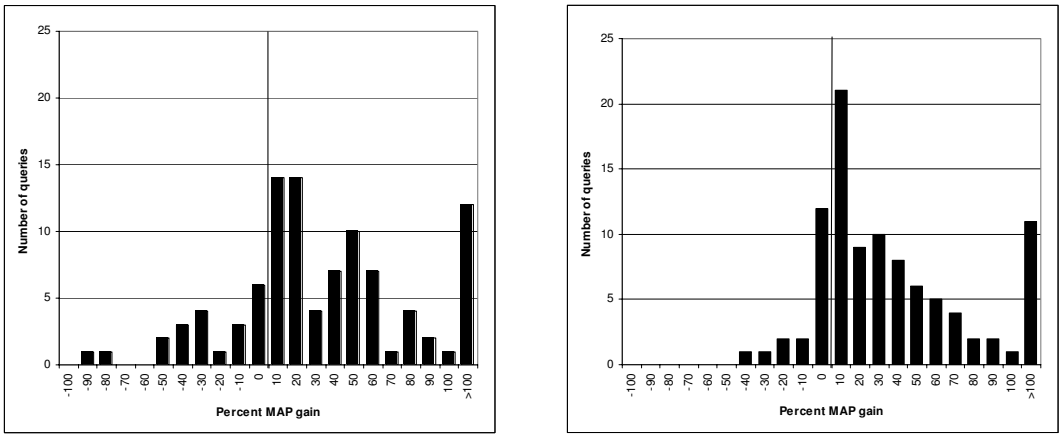
Chapter 1

Introduction

The traditional view of an information retrieval algorithm has been of a static process that takes some input observation, such as a user's query, and produces an output, such as a set of ranked documents from a collection. This thesis represents the first step in a new research direction: measuring and exploiting the *sensitivity* of an algorithm – how its output changes with small changes in inputs or parameters – to improve its performance. Essentially, we simulate the uncertainty inherent in the difficult matching problem the algorithm was created to solve.

The goal of this dissertation is to show that we can exploit this insight to develop robust, general-purpose algorithms for improving query expansion and related model estimation problems in information retrieval. By a robust algorithm, we mean one that not only produces good results on average, but is also likely to have good worst-case performance on any individual problem. We make substantial progress toward our goal with a novel application of two powerful techniques. First, we apply efficient types of *sampling* in new ways to obtain risk estimates for variables of interest in arbitrary retrieval models, and to smooth noise across combined language models. These risk estimates are then used to form a *convex optimization* program that solves the robust model estimation problem.

Past research efforts on information retrieval algorithms have focused largely on achieving good average performance, without much regard for the stability of individual retrieval results. The result is that current models are not robust and can still fail in basic ways, leading to poor results for individual queries. For example, current retrieval models often fail to retrieve documents that cover all aspects of interest that were implied by the query. This is reflected in the unpredictable benefits of current query expansion methods, in which



(a) Unstable baseline expansion (b) Robust version

Figure 1.1: Example showing state-of-the-art, but unstable, baseline expansion algorithm (left), compared to our goal of a robust version (right). Both methods achieve the same average MAP gain (30%), but the robust version does so with greatly reduced downside risk. Example shows results on the TREC 1&2 corpus.

accuracy can be greatly hurt by the automatic addition of irrelevant terms.

Figure 1.1 shows a concrete example of the instability of a current state-of-the-art query expansion algorithm (left). The histogram bins queries according to different levels of MAP gain or loss caused by applying the query expansion algorithm. Clearly, even the state-of-the-art algorithm still has unsatisfactory downside risk, as shown by the significant left-hand tail on the left-side histogram: for example, a number of queries at the extreme left experience 50% or more drop in MAP. Our goal is to eliminate or at least greatly reduce as much of this downside tail as possible, to obtain a *robust* version (right) that has greatly improved stability, but with average MAP at least as good as the baseline method. The histogram at left shows the Indri 2.2 expansion algorithm on the TREC 1&2 topics. (In fact, the histogram at right shows the actual performance of the optimization method we develop in Chapter 6.)

In addition, while state-of-the-art statistical retrieval models have recognized the importance of quantifying uncertainty, the practical implications of treating important entities such as queries and documents as random variables instead of single observations have not been fully explored. Thus, a further contribution of this thesis is to show how a random variable approach can result in useful new algorithms, including more precise word similarity measures and natural generalizations of existing query difficulty measures.

In solving the problem of estimating a good expansion model for a particular query, there are a number of competing objectives and constraints for the estimation problem: objectives such as maximizing expected utility of the model versus the risks of the multiple sources of evidence that the model is based on; the dependencies between the sources of evidence themselves; and additional important model constraints such computation cost and query aspect coverage. However, existing query expansion methods do a poor job of capturing these tradeoffs and goals in a unified, accessible framework.

To make progress on these problems, we bring together novel applications of two powerful techniques. First, we provide a sampling-based formulation of retrieval scoring and use this to estimate important quantities such as the mean and covariance of the output of an arbitrary feedback algorithm. The use of sampling fits well with our goal of general-purpose methods, because typically we need to assume very little about the functions being sampled. This allows our methods to handle arbitrarily complex retrieval operations and in general be applied in a broad family of retrieval scenarios.

Second, we introduce a novel general-purpose risk framework that characterizes query model estimation as a convex optimization problem. The objectives and constraints of the convex program are derived using the sampling-based estimates developed in the first part of the thesis. In this way, we can find query models that are optimal with respect to the tradeoffs between a number of competing optimization goals, in a way that would be difficult or impossible to specify with a single formula. Information about the solution can then be used as part of a retrieval algorithm. For example, a selective query expansion algorithm will not expand if there is no feasible solution to the optimization problem.

In the remainder of this chapter, we give some background into the general problem, describe why our research goal of flexible, robust retrieval algorithms is important, and describe the role of risk estimation. We then summarize some key challenges of this research, and the main methods that we use for achieving our goals. We close by summarizing the theoretical and practical contributions of this research.

1.1 The information retrieval problem

In a very broad sense, both people and computers often need information when performing particular tasks. We call the requirements that this information must satisfy the *information need*, and it might be only loosely defined, as might the tasks themselves. A query is a particular expression of an information need and may be an incomplete or vague description

of the information need (which we do not observe directly).

Information retrieval (IR) is a branch of computer science whose main goal is to provide effective methods for satisfying information needs. It has traditionally been distinguished from *database retrieval* by the fact that the representation of information is usually more loosely structured than the rigid table-based organization of a database, and that the information need is often not completely specified. However, the distinction between these two fields is becoming more blurry as more structure is used in documents. Information that satisfies an information need is called *relevant*.

The scope of information retrieval is as broad as information itself. Early IR research, covering a period roughly from the mid 1950s to the late 1970s, focused on text – especially text of interest to library applications such as books or journals. Today, however, retrieval algorithms of one kind or another are also applied to video, digital photos, scanned and on-line handwriting, genetic data, music, audio clips, and hypertext, not to mention the hundreds of different human languages handled by cross-lingual IR.

There is a general formulation of IR that all of these applications share. A retrieval algorithm is given a *query* generated by a user that represents their information need. In the case of text, this query consists of a series of words, along with possibly a set of relations between them. We assume that the information to be found resides in a *collection* which consists of a set of *documents*. Here the term document is very general and refers to a basic unit of information that could be a Web page, image, audio clip, and so on.

Given the query, the retrieval algorithm then scores the documents in the collection, ranking them according to some measure of how well the query terms and relations are matched by information in the document. For text, the relations most often used between terms are co-occurrence or proximity constraints. Traditional relevance also relies on the frequency with which terms occur in a document, and how unusual the terms are in the collection.

1.2 Why robust retrieval algorithms are important

In looking at how effective a retrieval algorithm is, it is important to distinguish its accuracy in the average case from its accuracy on individual queries. An algorithm might have good average accuracy, but have a large variation in accuracy from query-to-query, so that a few queries are satisfied with extremely high accuracy, but other queries obtain disasterously low results. Another algorithm with equally good average accuracy might be much more

consistent, avoiding the worst-case performance of the former method, while obtaining slightly worse results than the best-possible case of the more unstable algorithm.

Query expansion, for example, is a widely-used information retrieval technique that adds new words to a user's query in hopes of bridging the difference in vocabulary that might exist between relevant documents and the user's query. When information is available from a user about which documents are relevant , we can perform *relevance feedback* by expanding the query with terms from the relevant documents. If no relevance judgements are available, we can attempt to invent some, by assuming the top k documents are relevant: this operation is known as *pseudo-relevance feedback*, or *blind feedback*.

State-of-the-art feedback methods usually improve search accuracy on average, but can also significantly hurt performance for specific queries [Carpineto et al. 2001a]. A desirable goal is therefore to investigate more robust expansion algorithms that can reduce the number and severity of such failures without hurting overall precision. This is an important unsolved problem for current information retrieval research, and one significant reason why Web search engines still make little or no use of pseudo-relevance feedback.

Instability in retrieval is not desirable for a number of reasons. First, it leads to dissatisfied users, who typically prefer results that are reasonably good and predictable to results that are sometimes very good but completely unpredictable. Second, worst-case performance may be critical in retrieval applications that strongly emphasize precision over recall and thus have a low tolerance for noise. For example, given a student-oriented learning goal, a software language tutor might need to retrieve an appropriate example from a collection, or choose an alternative strategy if a high-quality example cannot be found, instead of showing the student a poor example.

We now give a short explanation for how the concept of risk is important to our goal of robust retrieval algorithms, and how uncertainty can be estimated and exploited to accomplish our research goals.

1.3 Estimating and exploiting risk in information retrieval

Current instability of retrieval algorithms is a result of an inevitable feature of information retrieval: *uncertainty*. First, a retrieval algorithm cannot know the queries that will be presented to it ahead of time, and even if it did, the user's information need may be vague or incompletely specified by these queries. Even if the query is perfectly specified, language in the collection documents is inherently complex and ambiguous and matching this against

the query is a formidable problem by itself.

The result is that many important quantities calculated by the retrieval system, whether a relevance score for a document, or a weight for a query expansion term, should not be seen as fixed values, but rather as random variables whose true value is uncertain but where the uncertainty about the true value may be quantified by replacing the fixed value with a probability distribution over possible values. It seems evident that any algorithm that hopes to be robust would do well to include estimates of uncertainty, such as probability distributions, as one factor in its internal calculations, so that it can quantify the risk or uncertainty associated with its output. For example, a query expansion algorithm should be able to control the trade-off between using a group of reliable, but possibly less effective expansion terms compared to a number of more unusual high risk, high reward terms.

1.3.1 Estimating risk and reward

The tradeoff between risk and reward is a familiar dilemma from everyday life. When we need to make an important decision, we often don't know all the facts with certainty. Instead, we must first understand how uncertain we are about the facts, and then based on how the facts might vary, estimate a range of possible outcomes, including best- and worst-case scenarios[1].

Similar ideas – in a more mathematically rigorous way – can be applied to the problem of searching for information. The decision by an algorithm to return a document to a user is typically taken under great uncertainty about the true nature of the user's needs, the language of the document, how well the query matches a document, and so on.

In order to consider factors such as risk and reward, a retrieval algorithm must have ways to quantify them somehow. Existing retrieval algorithms have focused almost exclusively on the 'reward' aspect of retrieval, which is typically quantified in statistical models by the probability of relevance given a document and query. Far less research has examined the critical additional aspects of 'risk' in a systematic way.

One reason having estimates of risk is important is that, as we show in this thesis, such estimates allow us to improve the robustness, or worst-case performance, of our algorithms. For example, when a query expansion algorithm detects a situation where its proposed query model is highly uncertain, it can scale back to a more conservative strategy that gives the original safe query terms much more weight. As we show later in Chapter 6, this

[1] Here we use the word *risk* in an informal sense without any specific mathematical definition assumed.

behavior, known as *selective query expansion*, is less effective when estimates of term risk are ignored.

One practical way to quantify uncertainty in a variable is to estimate a probability distribution for that variable. We can then make assumptions that certain properties of that distribution, such as the *variance* (or covariance in several variables) are an acceptable proxy for our uncertainty about the variable. If this can be applied to estimate uncertainty in the input to an algorithm, then that algorithm may be able to quantify the uncertainty in its output by somehow propagating the input's statistical properties through the complex system. From this new information, the algorithm can obtain confidence intervals on likely outcomes, which we show can be used for improved decision making, algorithm calibration, and model combination.

1.3.2 Exploiting information about risk

We now give a simple hypothetical example of how adding information about risk, in the form of covariance, can improve model estimation.

Suppose the task is to estimate the words that are likely to occur in relevant documents, based on a query string observed from the user: "parkinson's disease". A baseline algorithm estimates that some related words (with their estimated probabilities of relevance) are "disorders" 0.06, "syndrome" 0.05, and "brain" 0.04. Looking at just these relevance scores alone, an algorithm has no information about the meaning of the words – for example, that "disorders" and "syndrome" express similar concepts – or how the words co-vary, or how confident we are in these individual scores. Therefore, if a query expansion algorithm had to pick the two "best" related words simply on the basis of relevance scores, it would pick "disorders" and "syndrome". More generally, if the algorithm had 100 words, it might pick the k highest-scoring words, or employ some threshold. This is largely how current query expansion methods operate.

Now suppose we have new information: we know that "disorders" and "syndrome" are likely to be highly correlated with each other in documents. However, we are still aware that "brain" has almost as high a relevance weight as the other two, meaning it is still likely to be a distinctive term in some relevant documents. In this case, the best two related words to choose may no longer be "disorders" and "syndrome", because we gain little advantage from choosing features that select the same feature in relevant documents twice: essentially, a redundant bet. The pair "syndrome" and "brain" may be a better choice

because each is correlated with a different query term, and there is also some relation of each to both terms: informally, they "cover" the meaning of the query better than two disease-related terms. Thus, in some cases it makes sense *not* to choose the term with the highest relevance weight. This example is simplistic but shows how information about how terms in a relevance model co-vary can strongly affect how a model is estimated[2].

1.3.3 Using sampling to estimate risk

There are several challenges in attempting to quantify risk. Important random variables, such as a set of query expansion term weights, may be the output from complex non-linear functions that are difficult to approximate. It is also not clear how accurate our estimates need to be for them to be useful. Finally, if we need to ask the retrieval system for additional information, this requires extra computation, which should be minimized for the algorithm to remain practical.

In this dissertation the main tool we use to tackle these issues is *sampling*. This includes novel applications of some powerful sampling techniques from other fields, such as the unscented transform developed for particle filtering.

We adapt a Bayesian inference framework in which pluggable modules can work together using the shared mechanism of probability. Bayesian inference also gives principled ways to include prior knowledge about a given problem, to adjust a model in response to new evidence, and to combine evidence from multiple hypotheses.

In a Bayesian formulation of retrieval, the user's query represents evidence about relevance, and is used to update the parameters of a model that describes what relevant documents look like, or perhaps how they differ from non-relevant documents. To calculate a document score (for example) based on our retrieval model that uses uncertain parameters, the correct formulation in Bayesian statistics is to calculate the expected outcome of the model over all possible values of the parameters ([Duda et al. 2001], p. 487). This means that when we account for uncertainty, formulas such as document scores become *integrals* over the space of parameters in the model. Unfortunately, such integrals are usually very expensive to compute, which makes their direct use impractical for real-world IR systems. There are, however, numerical techniques for efficiently approximating these integrals by evaluating the function at different points in the parameter domain and then combining the

[2] In fact, in Chapter 6 we show how an optimization approach produces exactly this type of term selection behavior, using the same *parkinson's disease* query in Figure 6.5 as an example.

resulting function values.

One such method is *Monte Carlo integration* which uses sampling to evaluate the integrand in regions where it is likely to take on large values or to vary rapidly. The use of sampling allows us to use a rich set of potential document and term scoring functions, because almost no assumptions are made about the nature of the the integrand, which is treated like a 'black box'. Monte Carlo sampling also provides a way to address robustness, because instead of calculating one estimate of how relevant a document is, we can sample several different estimates (using the right choice of sampling distributions) and combine the results. In statistical terms, using multiple samples and sampling methods can reduce the variance of our estimate of the true value. For efficiency, we tend to emphasize *deterministic* sampling methods that use a small number of samples.

In a different setting, the idea of taking a representative sample will also prove to be a powerful idea for getting an accurate estimate of the sensitivity of a feedback algorithm, and for smoothing out the performance of an unstable feedback algorithm using *bootstrap sampling*, which is described further in Chapter 3.

1.3.4 Applying risk estimates to finding optimal models

In an optimization approach to model estimation, instead of trying to solve a model estimation problem by finding an explicit formula for the parameters, we take a more flexible path: we specify objective and constraint functions that the ideal solution should satisfy, and then the actual work of searching the parameter space for the optimal solution is performed by a general-purpose *solver* routine.

Using the "parkinson's disease" example of Section 1.3.2 above, we can create an optimization model that instructs the solver to search for sets of terms satisfying two simultaneous criteria. First, it should prefer sets of terms that have high relevance weights; this is the "reward" criterion to maximize. Second, the solver should prefer sets of terms that are minimally "redundant": this is a "risk" criterion to be minimized. The result will be a method of model estimation in which the tradeoff can be easily adjusted. As we show in Chapter 6, such risk constraints help stabilize the performance of model estimators. Typically, our optimization will embody a basic tradeoff between wanting to use evidence that has strong expected relevance, such as highly-ranked documents, or highly-weighted expansion terms, and the risk or confidence in using that evidence.

If the solver's search of the model parameter space were not efficient, there would be

little incentive to apply this approach to the time-critical task of retrieving documents from a large collection. However, recent advances in the speed of both hardware and interior-point solver algorithms are making convex optimization a readily applicable technology that can quickly handle problems having hundreds or thousands of variables. Part of our aim in this dissertation, therefore, is to introduce the benefits of this important tool to information retrieval problems that might otherwise have been too complex or difficult to solve until now. In Chapter 6 we give an in-depth treatment of the convex optimization approach and its other advantages.

1.4 Challenges to be addressed

In attempting to create improved models and algorithms for information retrieval, the following challenges must be addressed along the way and will shape the solutions we choose. Because IR typically involves human language, some of these challenges are inherited from the general problem of attempting to capture meaning and interaction in language.

High dimensionality. Information retrieval models typically involve query and document representations having thousands of dimensions. It is therefore important that the mathematical techniques we use can scale well, both in terms of accuracy and efficiency, to a large number of dimensions and a potentially large number of parameters.

Run-time efficiency. Since a search system must respond to a user within a few seconds, algorithms for analyzing a query, or scoring documents against a query, must be extremely efficient. Our ability to use more sophisticated models of language for IR is thus restricted to some extent by our ability to compute with them efficiently. Thus, if simple closed-form solutions are not available, methods for fast approximations or precomputation become very important.

General-purpose methods. A well-designed retrieval system is modular: the implementations of different subcomponents of the retrieval process, such as term weighting, query expansion, and document/query matching functions may be replaced or modified. Some of this flexibility may even be available to the user, via a more advanced query language for example. When we need to analyze the performance of a particular subcomponent (such as the sensitivity of a document score or a term's relevance weight), we would like to use methods that treat the subcomponent as a 'black box' and make as few assumptions about the nature of the implementation as possible. In this way, we ensure that our model estimation methods can be applied in as broad a range of scenarios as possible.

Use of training data. One relatively recent development for information retrieval is the availability of large datasets for training and analysis. This includes document collections (particularly the Web), query logs from Web search engines and standardized test collections, such as from TREC. We would like to take advantage of these resources to help train our statistical models, by using empirical Bayes methods to estimate effective prior probability distributions. In query expansion scenarios, on the other hand, human-labeled data is typically not available, or extremely limited. It is therefore also important that we understand how sensitive our models are to having only a small amount of training data from which to learn.

1.5 The problem of query drift

Central to the problem of unstable query expansion algorithms is the problem of *query drift*. Query drift is the change in focus of a search topic away from the original intent of the user, typically because of incorrect or incomplete query expansion or feedback methods. The nature and causes of query drift have been examined in a number of studies [Mitra et al. 1998] [Harman & Buckley 2004], and form an important basis for development of solutions later in this thesis. The main causes of query drift can be categorized as follows.

Poor initial retrieval. A key assumption of feedback methods is that at least some of the top-ranked documents are relevant, on average. Therefore, one of the most common scenarios causing query drift is a lack of relevant documents in the top-retrieved documents.

One attempt to reduce query drift has focused on improving the precision of the top-retrieved documents. Mitra, Singhal, and Buckley [Mitra et al. 1998] perform re-ranking by finding relevance indicators that enforce conditions such as boolean term combinations and term proximity, while also rewarding concept diversity. Crouch *et al.* [Crouch et al. 2002] perform a similar type of reranking but focus on matching unstemmed query terms in documents using heuristics such as sums of query term weights.

Another method for improving the chance of finding at least some relevant documents is to extend the initial retrieval to use multiple alternate query hypotheses. Our use of query variants in Chapter 3 and the AbraQ algorithm [Crabtree et al. 2007] are two examples of this approach. Kurland, Lee, and Domshlak [Kurland et al. 2005] also employ multiple query hypotheses in the form of pseudo-queries: "Starting from the original query, our methods repeatedly seek potentially good renderers of a current set of pseudo-queries,

guided by the hypothesis that documents that are the best renderers of a pseudo-query may be good alternate renditions of it." Critically, they use a re-anchoring strategy at each iteration to interpolate the original query score with new relevance scores. Kurland *et al.* also use *document clustering* to find related documents to those in the initial set(s).

Poor coverage of query aspects. No matter how many or few relevant documents are found in the initial retrieval, recent studies [Harman & Buckley 2004] [Collins-Thompson & Callan 2005] have recognized that weak *aspect coverage* in the resulting feedback model or final expansion model is something algorithms must detect and remedy to avoid query drift. Here we use an informal definition of *aspect* to mean a topic of interest implied by all or part of the query. For example, a query such as *economic impact of recycling tires* would have as possible aspects the concept areas of *recycling*, *economy*, and *tires*. A more specific aspect might be *economic impact*.

Certainly, poor initial retrieval is likely to lead to poor aspect coverage. Yet excellent initial retrieval may also suffer from unbalanced aspect coverage in current feedback algorithms. In a summary of the results from the 2003 Reliable Information Access Workshop, Buckley writes, "...relationships between aspects of a topic are not especially important for state-of-the-art systems; the systems are failing at a much more basic level where the top-retrieved documents are not reflecting some aspect at all. " [Buckley 2004]

In ad-hoc retrieval, the approach of Mitra *et al.* includes a document re-ranking function that rewards multiple *independent* concepts based on word co-occurrence in the top 1000 documents. Their heuristic formula downweighting the contribution of correlated terms obtained consistent improvement compared to not using word association.

Also related to aspect coverage in feedback models is the problem of *subtopic retrieval* in which the criterion for selecting the top documents looks beyond the assumption of independent relevance of documents, to select a set of documents that together cover a set of aspects or subtopics [Zhai et al. 2003]. Here, the emphasis is on the results presented to the user, rather than the features learned for a feedback model.

Detecting poor or uncertain aspect coverage is closely connected with algorithms for estimating query difficulty, which we discuss in more detail in Section 4.6.5.

Noise terms in feedback model. Selecting feedback terms is typically done in a two-step process. First, a score is assigned to each term. This score often has the form of a two-part scheme that combines some measure of the term's rarity with its likelihood of being in the

top-ranked documents. In essence, a score should reflect high probability of being in a relevant document, while also being a good discriminator against non-relevant documents. The widely-used *tf.idf* scheme is one example of a scoring formula that combines these two factors. In the second step, a greedy selection method takes the top k terms using either a rank threshold (top 10) or a score threshold (all terms with score $>= S$).

Ideally, a term will achieve a high S because both *tf* and *idf* components have high values. This corresponds to a rare term occurring very frequently in the top-ranked documents. In many situations, however, one artifact of a *tf.idf* method is that a term can still achieve a high score when only one component, such as *tf*, is very high. This is the case with stopwords, and in fact this scoring behavior becomes evident when the use of a stoplist is turned off – even with feedback algorithms found in more sophisticated retrieval systems, such as the Indri search engine [Strohman et al. 2004] used in this thesis.

We explore this phenomenon further in Section 3.4.10 and show that techniques like *bagging* can help reduce the noise from unstable feedback term weighting schemes by finding the terms that have more consistent scores under multiple related hypotheses.

The value in improving feedback algorithms is not restricted to applications using either explicit user input or completely automatic methods. To accomplish personalization of search results, data can be provided *implicitly* in a relevance feedback framework by using user data. The resulting feedback model is then used to re-rank documents with a bias toward user interest. This approach is introduced by [Teevan et al. 2005]. In essence, the main goal is estimating a more accurate model of relevance, and so feedback methods can be seen as more than just an add-on component of a system, but an integral part of the scoring procedure itself.

1.6 Summary of original contributions

This dissertation introduces new theoretical models, new statistical methods, and new retrieval algorithms that are enabled by these models and methods. Our main contribution is the development of robust model estimation methods in information retrieval. The key property of these algorithms is that they have significantly better worst-case performance than current methods, with no reduction – and in many cases, significant improvement – in average-case performance. The following three subsections summarize the theoretical, statistical, and algorithmic contributions in more detail.

1.6.1 Risk estimation for information retrieval

A principled risk framework for query model estimation. We frame the problem of estimating an optimal query model as a convex optimization problem. Traditional methods for estimating query models have made fairly restrictive assumptions in order to simplify the problem. For example, many traditional methods for feedback, such as Rocchio, estimate each term parameter independently of the others. Such methods fail to capture important constraints involving the *entire set* of terms, such as the totality of query aspects being covered. By formulating the problem in terms of convex optimization problems we can extend our search space beyond a greedy threshold approach, to find optimal subsets of terms with respect to such set-based conditions. Our framework is quite general in that it can support any situation in which there are multiple sources of information about relevance.

A sampling-based view for multi-strategy retrieval. We extend existing approaches to statistical retrieval with a theoretical framework that proposes a novel view of document scoring as the combination of multiple sampling distributions of the score integral. Each sampling distribution corresponds to a different retrieval strategy. Existing methods from Monte Carlo integration are then used to perform model combination for these multiple complementary strategies. Essentially, sampling is used as a way to create related retrieval problems whose results may then be compared and merged. This formulation also gives new insights into existing document scoring formulas. For example, we show that the document-based scoring formula in the Relevance Model [Lavrenko 2004] is actually a special case of a Monte Carlo integration heuristic called multiple importance sampling [Veach 1997]. This in turn suggests useful new generalizations of the Relevance Model scoring methods.

New IR evaluation methods. We introduce a new family of measures, called *R-Loss* measures, that quantify expansion algorithm risk/variance. With these measures, we then construct new types of risk-reward curves to compare query expansion algorithms. Using a novel analogy between information retrieval and computational finance, we also obtain useful new summary statistics such as the *midpoint risk tradeoff*, and find counterparts to the important *two-fund theorem* from finance to derive a new heuristic for finding an optimal query interpolation parameter.

1.6.2 General purpose statistical methods

Data perturbation kernels. To solve the problem of finding effective query-specific similarity measures, we introduce a general family of kernels called *data perturbation kernels* that efficiently learn a metric over the input domain based on a small number of carefully chosen perturbations of the training set. We apply this to information retrieval by treating the query as a very small training set, and create a small number of auxiliary queries with different relative term weights compared to the original query. In this way, two input points, such as words, are considered similar if their probabilities have similar sensitivity to the same perturbations of the term weights in the original query.

Sensitivity analysis for retrieval algorithms. We describe a new technique for computing the sensitivity of arbitrary retrieval functions, such as document scoring functions. The method uses a novel application of the *unscented transform* from particle filtering to give an accurate approximation of the first and second moments of arbitrary non-linear scoring and other retrieval functions. Instead of trying to approximate the potentially complex retrieval function, the unscented transform gives an algorithm for selecting deterministic representative samples in the input space. We assume a general parametric family for the input distribution: the logistic normal, which can approximate the Dirichlet as a special case. We generalize the *clarity* measure of query difficulty to add a clustering factor for the collection based on estimating the sensitivity of feedback models to perturbations of the query.

1.6.3 Robust model estimation algorithms

Stable pseudo-relevance feedback models. We employ a novel use of sampling to stabilize the language model estimated for pseudo-relevance feedback using a baseline algorithm. Inspired by traditional bagging, we use replacement sampling of the input data to obtain multiple predictors for the feedback model. The output, however, is a set of multi-dimensional vectors instead of single-valued numeric or class predictors. Thus, instead of simple averaging, we fit a latent Dirichlet distribution and find approximate maximum likelihood model parameters using fast quasi-Newton methods. The resulting feedback model is much less noisy than any of the individual input models, because it rewards terms that have consistently good weights across multiple samples, even if they are not the highest-weighted terms in any given individual input model. Using the combined feedback model

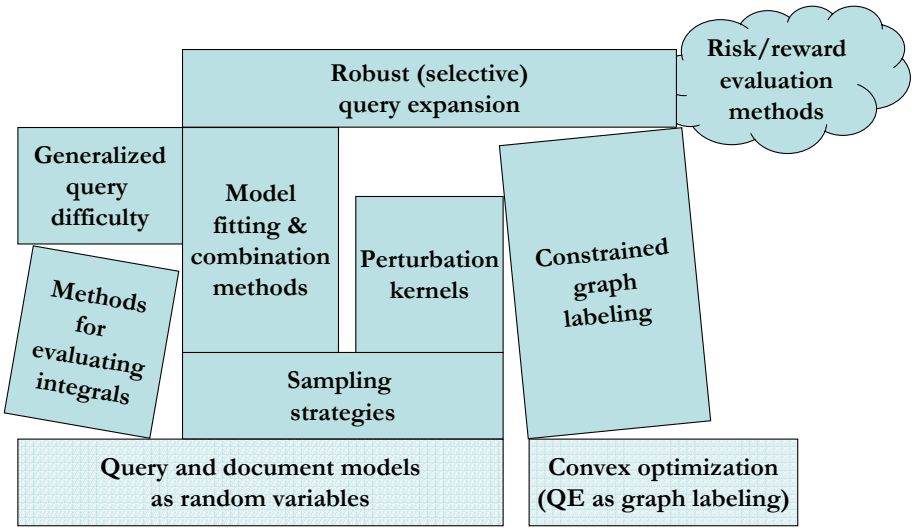


Figure 1.2: Overview of thesis components, with gravity showing dependencies.

results in small but consistent improvements in precision. In addition, it makes the use of a stopword list much less critical – unlike with current feedback algorithms – because stopwords and noise terms tend to be high-variance features in the combined model which are then automatically removed. Since our method treats the baseline model estimator as a black box it is very general and can be applied to improve arbitrary feedback algorithms.

Robust selective query expansion. We present a new algorithmic framework for robust query expansion that treats queries and top-ranked document sets as random variables and 'wraps' a baseline expansion algorithm. Starting with a small number of query variants, we learn stable individual feedback models and a perturbation kernel, and perform model combination using either a heuristic approach (Chap. 3) or a convex optimization approach (Chap. 6). Both methods result in improved robustness, while the latter method also expands selectively for risky queries in a principled way, and is highly resistant to noise in the baseline feedback algorithm.

1.7 Overview of thesis organization

A graphical view of the thesis components and dependencies between them (in the style of [Karger 1994]) is shown in Figure 1.2. The first two chapters summarize our research and introduce basic problems and concepts. Chapter 2 begins with a short review of some probability theory and Bayesian decision theory, which are the foundation of statistical

methods in this thesis. It then goes on to describe two more basic concepts: sampling and Monte Carlo integration. As an application to information retrieval, the Relevance Model is discussed and formulated in terms of Monte Carlo integration. In Chapter 3, we describe a framework for pseudo-relevance feedback and specific implementations of robust query and document model estimators that are based on sampling. We also describe risk-reward curves, a new evaluation method for feedback algorithms. Data perturbation kernels are introduced in Chapter 4 to learn query-specific similarity measures. We also use these kernels to generalize a class of query difficulty measures. Chapters 5 and 6 bring together the estimation methods from previous chapters within a novel risk framework based on convex optimization to solve the problem of finding robust query models. We review basics of convex optimization in Chapter 5 and then in Chapter 6 describe and evaluate objective functions and constraints that are useful for query model estimation. This includes the ability to constrain solutions by aspect coverage or computation cost. Chapter 7 summarizes our contributions in detail and discusses new research directions enabled by this work.

1.8 Summary

This chapter explained and motivated the goal of this research, which is to create robust, general-purpose algorithms for model estimation in information retrieval. We described a number of problems with current model estimation methods, such as their instability for individual queries and their inability to capture the tradeoffs between a wide range of possible objectives in a principled way.

To achieve our goal with these challenges in mind, we apply the union of two powerful techniques. First, we apply sampling to obtain more robust estimates for important quantities in information retrieval models such as the feedback model from initial top-retrieved documents. Our methods make few assumptions about the details of the baseline retrieval method. This sampling-based formulation leads to new insights and algorithms for retrieval, including novel ways to estimate good query variants and more precise language models for pseudo-relevance feedback.

Second, we apply the new information gained from sampling, such as covariance matrices, to create effective additional constraints on traditional retrieval algorithms that improve their robustness. To do this, we introduce a novel risk framework that treats model estimation as a convex optimization problem. One result of this is a new, principled algorithm

for selective query expansion that is sensitive to the risk of individual query scenarios: if there is no feasible solution to the optimization problem, we do not attempt to expand the original query. Useful additional model constraints such as robustness, aspect coverage, and sparsity can be expressed within this framework to give a very flexible general-purpose approach to finding optimal query models in a variety of useful retrieval scenarios.

We believe this is the first significant exploration of sampling methods for estimating information retrieval models, and the first general query expansion framework based on convex optimization for information retrieval problems of any kind.

Chapter 2

Sampling Methods for Information Retrieval

In this chapter we introduce some basic statistical and sampling techniques and terminology used throughout the rest of this dissertation. We then focus on sampling-based methods for calculating expectations and other integrals, and explain such methods help us achieve our goal of flexible, robust information retrieval algorithms. As an example application, we describe the Generative Relevance Model [Lavrenko 2004] (GRM) and show how a Monte Carlo-like formulation of score estimation in the generative relevance model leads to new insights into document scoring. We show that document-based ranking in the GRM may be seen as a special case of Monte Carlo integration using a sampling technique known as the balance heuristic. This in turn suggests new algorithms for robust query expansion. We next discuss the general issue of uncertainty in information retrieval and give some background on sampling and its use for retrieval problems.

2.1 Uncertainty and Risk in Information Retrieval

As we discussed in Chapter 1, uncertainty is an inherent feature of information retrieval. To achieve our goal of robust retrieval algorithms, we need ways of quantifying uncertainty so that retrieval algorithms can include it as a factor in their calculations.

We take the view that many of the quantities that appear in retrieval models, such as the term weights assigned for query expansion, the final query-document score, and the ranking of top-retrieved documents, are more properly treated as *random variables* instead

of single observations. For example, in the case of a query from the user, we treat the observed sequence of words $\boldsymbol{q}$ as a noisy version of a hidden 'true' information need $\boldsymbol{q}'$ through a perturbation or translation process $p(\boldsymbol{q}|\boldsymbol{q}')$. In this view, the scoring function $\Delta(\boldsymbol{q}, \boldsymbol{d})$ comparing a document $\boldsymbol{d}$ against $\boldsymbol{q}$ is theoretically not a single-point comparison of $\boldsymbol{q}$ and $\boldsymbol{d}$, but an *expectation* over a density $p(\boldsymbol{q}')$ derived from $\boldsymbol{q}$.

$$\Delta(\boldsymbol{q}, \boldsymbol{d}) = \int d(\boldsymbol{d}, \boldsymbol{q}')p(\boldsymbol{q}'|\boldsymbol{q})d\boldsymbol{q}' \tag{2.1}$$

This by itself is not a new theoretical idea. Recent statistical frameworks for retrieval, such as the language modeling approach [Ponte & Croft 1998], the Relevance Model [Lavrenko 2004] and the risk minimization framework [Lafferty & Zhai 2001] have recognized the importance of quantifying uncertainty in retrieval models. Typically, this is done through the use of a probabilistic approach, and in particular a Bayesian methodology that provides a principled way to estimate the posterior distributions of important quantities, given some observed evidence such a query from the user.

In practice, however, even state-of-the-art algorithms have only begun to explore the power and generality of such Bayesian frameworks. There are several reasons for this. First, these formulations are relatively recent and simple approximations have been the most productive to explore first. Second, more advanced applications of the models are more computationally demanding to calculate. For example, in theory we must integrate complex integrals over large-dimensional parameter spaces. Part of our work in this thesis will explore ways to mitigate this expense, using efficient sampling methods. We give further details on how a sampling approach can lend insight into Bayesian retrieval models using the Relevance Model as an example in Section 2.4.1. First, we give some background on basic statistical methods.

2.2 Background on Sampling

An essential tool that we will use to accomplish our goals of both flexibility and robustness is *sampling*. Sampling is the process of generating observations of a random variable using the probability density defined for that variable. In our work, the random variables are typically parameters. These parameters usually occur as part of a retrieval method, such as the document weights used for pseudo-relevance feedback, or define generative models of text, such as language models for queries and documents.

Sampling is an effective, flexible approach for several reasons. First, the use of sampling allows us to vary the input to any retrieval process $\boldsymbol{F}$ in a principled way, by encoding the nature of our uncertainty about the data in the sampling distribution. We can treat the the process in question as a black box, so that instead of having to model the internal parameters of $\boldsymbol{F}$, we vary the input to $\boldsymbol{F}$ directly by sampling from its probability density. This allows us to make very few assumptions about the nature of the retrieval process being analyzed.[1]

The notion that documents, queries and other objects of interest in IR are samples from probability distributions is a fundamental concept in probabilistic approaches to IR such as the language modeling approach [Ponte & Croft 1998] and the Generative Relevance Model [Lavrenko 2004]. Thus, the idea of applying sampling methods fits naturally with these types of retrieval models.

Sampling is also important to consider for efficiency reasons. In Bayesian frameworks for probabilistic IR, document scoring is formulated in terms of integration over a parameter space of query and document models. These models are often parameterized with respect to a large vocabulary, and thus the integrands may be high-dimensional, complex functions defined by the product of a large number of factors whose relative importance is not known in advance. For example, to score a document against a query in the risk minimization framework [Zhai & Lafferty 2006], the theoretical document score function for a document $\boldsymbol{d}$, query $\boldsymbol{q}$ having respective models θ_D and θ_Q for user $\mathcal{U}$ is

$$r(\boldsymbol{d}|\boldsymbol{q},\mathcal{U}) \propto \int_{\theta_Q}\int_{\theta_D} \Delta(\theta_Q,\theta_D)p(\theta_Q|\boldsymbol{q},\mathcal{U})p(\theta_D|\boldsymbol{d})d\theta_D d\theta_Q \tag{2.2}$$

In practice, one effective approximation for an integral having a high-dimensional posterior distribution $p(\theta)$ as a factor in the integrand is to simply evaluate the integrand at the mode $\hat{\theta}$ of $p(\theta)$, giving which simplifies the integral to

$$r(\boldsymbol{d}|\boldsymbol{q},\mathcal{U}) \propto \Delta(\hat{\theta_D},\hat{\theta_Q}) \tag{2.3}$$

This can be seen as a Monte Carlo-like estimate using a single sample. As we show later in this chapter, there are natural generalizations to scoring methods that combine multiple

[1] From a multi-task learning perspective, sampling is a way to create related problems: similar predictions on the related problems (inputs) define a similarity measure on the input space that can help with inductive transfer, e.g. label propogation in unsupervised learning.

samples. In general, it can be far more efficient to approximate the result of a process by averaging over a few samples from its output than to search for an almost exact answer in a large solution space. Sampling may sometimes be the only way to perform the estimate if the form of the process is complex and non-linear.

Sampling is relevant to the problem of robustness, because *model combination* among complementary retrieval strategies can be performed by an appropriate choice of sample weighting scheme. This is a novel and fruitful connection for information retrieval. In recent work, several studies have examined how to combine results of multiple related document and query representations. These typically involve creating several modified versions of the original query and combining the results. For example, to estimate query difficulty YomTov et al. [YomTov et al. 2005] created a set of subqueries by selecting terms one-at-a-time from the original query and combining the document rankings returned from each subquery. Ando et al. [Ando et al. 2006] performed query expansion by leaving out terms one-at-a-time from the original query to create a set of subqueries. They obtained scores for expansion term candidates by combining the term scores over the resulting sets of ranked documents. To our knowledge, no general model has been proposed that captures the similar nature of these various related applications and that can answer questions such as when and what types of subquery generation strategies are likely to be effective; how the results of the different subqueries (either document or term scores) should be weighted and combined; and the likely effect of combining different subquery strategies on retrieval accuracy and robustness.

We believe that our novel approach of combining a language model approach to IR with sampling methods provides an effective, simple and principled framework for addressing these types of questions.

2.3 Probability and statistics basics

Before describing our sampling framework, we give a brief review of some important concepts from probability theory and statistics that will be used in the rest of this dissertation. See [Pittman 1993] for further background on probability.

2.3.1 Probability density functions

The *cumulative distribution function* of a random variable $X \in \mathbb{R}$ is defined as

$$P(x) = Pr\{X \leq x\} \tag{2.4}$$

with corresponding *probability density function*

$$p(x) = \frac{dP}{dx}(x) \tag{2.5}$$

The function $p(x)$ is also known as the *density function* or the *pdf* of X. In the multidimensional case when we have a vector of random variables $(X^1, \ldots, X^m)$ the *joint cumulative distribution function* is given by

$$P(x^1, \ldots, x^m) = Pr\{X^i \leq x^i : i = 1, \ldots, m\} \tag{2.6}$$

with *joint density function*

$$p(x^1, \ldots, x^m) = \frac{\partial^m P}{\partial x^1 \ldots \partial x^m}(x^1, \ldots, x^m) \tag{2.7}$$

We then have

$$Pr\{x \in D\} = \int_D p(x^1, \ldots, x^m) dx^1 \ldots dx^m \tag{2.8}$$

for any subset $D \subset \mathbb{R}^m$ that is Lebesgue-measurable[2].

We can generalize this when the random variable X takes values in some arbitrary domain Ω to define the *probability distribution* or simply the *distribution* P of X as follows:

$$P(D) = Pr\{X \in D\} \tag{2.9}$$

for any measurable set $D \subset \Omega$, with $P(\Omega) = 1$. The density function is then the function p that satisfies

$$P(D) = \int_D p(x) d\mu(x) \tag{2.10}$$

[2]See, for example, background material at `http://en.wikipedia.org/wiki/Lebesgue_measure`.

2.3.2 Expected value and variance

The *expected value* or *expectation* of a random variable $Y = f(X)$ is defined as

$$\mathbb{E}[Y] = \int_{\Omega} f(x)p(x)d\mu(x), \tag{2.11}$$

and its *variance* is

$$V[Y] = \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right]. \tag{2.12}$$

The *standard deviation* of the random variable Y is denoted

$$\sigma[Y] = \sqrt{V[Y]} \tag{2.13}$$

We will assume that the expectation and variance of every random variable exist, that is, have a bounded integral.

2.3.3 Bayesian decision theory

We assume we can quantify uncertainty about values of specific parameters or variables via a joint distribution over those values. We can then quantify tradeoffs between decisions using these probabilities, combined with estimates of the costs that accompany decisions. This formulation is known as *Bayesian decision theory* and forms the theoretical basis for taking an action under uncertainty.

More formally, given a set of possible actions $\mathcal{A} = \{\boldsymbol{a_i}\}, i = 1, \ldots, m$ and a parameter space Θ, the penalty of taking action a_i for a specific parameter value $\theta \in \Theta$ is given by the *loss function* $\Delta(a_i, \theta)$. Taking an equivalent positive instead of a negative point-of-view, we may also refer to the *utility function* $U(a_i, \theta)$ to denote the *benefit* obtained by choosing action a_i for parameter θ. The *risk* or *expected loss* of a given action a_i is the expectation of $\Delta(a_i, \theta)$ over all possible parameter values, with respect to the posterior distribution $p(\theta|X)$.

$$R(a_i \mid X) = \int_{\Theta} \Delta(a_i, \theta)p(\theta|X)d\theta \tag{2.14}$$

The *Bayes optimal decision criterion* is to choose the action $a_\star$ that minimizes the expected loss over the posterior.

We denote by X_i an observation from the random variable X. We call X_i a *sample* from X, and we usually denote the number of samples, which we call the *sample size*, by N.

2.4 Monte Carlo integration

Because many important quantities in information retrieval models, such as document scores, are formulated as integrals in a Bayesian framework, it makes sense to look at methods for computing these efficiently if we are to use them in later risk frameworks. As a specific example from information retrieval, we describe the Generative Relevance Model and its use of integrals.

Monte Carlo integration is a widely-used method for evaluating integrals, particularly when the integrand is complex and the domain of integration is high-dimensional. A good overview of Monte Carlo integration is given in Kalos and Whitlock [Kalos & Whitlock 1986] and in Chapter 2 of Veach [Veach 1997].

First, note that the basic approach to evaluate a general integral of the form

$$\mathcal{I} = \int_{\Theta} f(\theta) d\mu(\theta) \tag{2.15}$$

on the domain Θ with measure $d\mu$ is to independently sample N points $X_1, \ldots X_N$ in Θ according to some density function $p(x)$, and then compute the random variable

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{2.16}$$

The random variable F_N is called a *Monte Carlo estimator* for $\mathcal{I}$. It is easy to show that F_N in Equation 2.16 gives an unbiased estimate of $\mathcal{I}$. We therefore focus on a method for reducing the variance of F_N, since variance determines the number of samples needed to get an accurate estimate of the integral. Large variance implies greater likelihood of inaccurate values for the estimate of $\mathcal{I}$, especially when the number of samples N is small. We emphasize that in addition to sampling methods that use randomness, we also consider several *deterministic* methods of choosing samples in Section 2.5.5.

In this chapter, we use the following notation. The integrand of interest is denoted $f(\cdot)$. A sampling technique $\mathcal{S}_i$ has sampling distribution $p_i(\cdot)$, and the number of samples taken from it is n_i. The j-th sample from $p_i(\cdot)$ is denoted by $X_{i,j}$. We denote a space of parameters (e.g. for a relevance model) as Θ. To integrate over a domain Θ, we need to specify a *measure* on Θ, denoted $d\mu(\theta)$. For this chapter, we use a measure $dp(\theta)$ that can simply be thought of as a probability density over Θ. In most of the cases we consider, this measure

is based on an observed query $\boldsymbol{q}$, in which case it is denoted $dp_{\boldsymbol{q}}(\theta)$.

Monte Carlo integration is simple and quite general. To implement it, there are only two operations needed: sampling over the domain, and function evaluation at a point in the domain. Prior knowledge about the nature of the integrand can be easily incorporated into the sampling technique to reduce the variance of the estimate.

2.4.1 The Generative Relevance Model

For a specific example of how a sampling approach can lend insight into Bayesian information retrieval methods, we now describe the Generative Relevance Model (GRM) introduced by Lavrenko [Lavrenko 2004], and how some important integrals that arise in that model are estimated. For example, we show how the standard GRM document scoring function can be seen as a type of importance sampling procedure using multiple relevance hypotheses.

In the Generative Relevance Model (GRM) both a document and a query are hypothesized to be samples from a shared generative relevance model (but with potentially different sampling functions for each). Let $\mathcal{R}$ be a binary random variable for which $\mathcal{R} = 1$ denotes relevance and $\mathcal{R} = 0$ denotes non-relevance. If we denote a joint probability distribution over documents and queries as $P(\mathcal{D}, Q)$, then we define the relevance model $P_{\mathcal{R}}(\cdot) = P(\cdot|\mathcal{R} = 1)$. The ideal relevance model would be based on a mixture model of the relevant documents. Since the set of relevant documents is not known, however, we must estimate $P_{\mathcal{R}}$ based on the observed query and top-ranked documents.

Let R be the set of relevant documents. Assuming that documents and queries are conditionally independent given $\mathcal{R}$, the joint probability of observing a word w and a document set $\mathcal{D}$ given $\mathcal{R}$ is

$$P(w, \mathcal{D} \mid \mathcal{R}) = P(w \mid \mathcal{D})P(\mathcal{D} \mid \mathcal{R}) \tag{2.17}$$

and so taking the marginal over documents, this becomes

$$P(w \mid R) = \sum_{\mathcal{D} \in C} P(w \mid \mathcal{D})P(\mathcal{D} \mid R) \tag{2.18}$$

where C is the document collection. In Eq. 2.17 the prior probability of a document *not* in the relevant set is set to zero.

The GRM uses a Bayesian formulation in which $P_{\mathcal{R}}$ is assumed to have parameters θ assigned from the domain of all possible parameters Θ. We define an initial probability

measure $dp_0(\theta)$ that is a prior over Θ. After seeing the query, we update the measure to create a posterior measure $dp_{\boldsymbol{q}}(\theta)$ defined by

$$dp_{\boldsymbol{q}}(\theta) = \frac{P_{\mathcal{R}}(Q = \boldsymbol{q}|\theta)}{P_{\mathcal{R}}(Q = \boldsymbol{q})} dp_0(\theta) \tag{2.19}$$

The relevance of a document is then expressed as an integral over the parameter θ with measure $dp_{\boldsymbol{q}}(\theta)$ and the integrand $f(\cdot, \theta)$ defining the ranking criterion. We denote the score for a document $\boldsymbol{d}$ with respect to a given query $\boldsymbol{q}$ as $p(\boldsymbol{d}|\ \boldsymbol{q})$. The general form of this integral is

$$p(\boldsymbol{d}|\boldsymbol{q}) = \int_{\Theta} f(\boldsymbol{d}, \theta) dp_q(\theta) \tag{2.20}$$

For *document-based ranking*, we compare the probability p_θ of a document $\boldsymbol{d}$ to the probability p_C of the collection C, under a shared distribution p_θ with parameters θ. This gives the integral:

$$p(\boldsymbol{d}|\boldsymbol{q}) = \int_{\Theta} \frac{p_\theta(\boldsymbol{d})}{p_\theta(C)} dp_{\boldsymbol{q}}(\theta) \tag{2.21}$$

Alternatively, we can estimate separate generating distributions in Θ for the document and query separately and then compare these two distributions using a similarity function such as KL-divergence – a method known as *model-based ranking*. In this case, the document score is

$$p(\boldsymbol{d}|\boldsymbol{q}) = \int_{\Theta} \Delta(\theta, \theta_d) dp_q(\theta) \tag{2.22}$$

where θ_d is the empirical distribution for document $\boldsymbol{d}$ in Θ, and $\Delta(\cdot, \cdot)$ is the similarity function.

In either case, calculating the integral for each document would be computationally expensive. As we noted earlier, previous work has used an approximation to the integral, typically taking the form of a single-sample estimate, where the sample is obtained by picking the most likely model. In the GRM itself, Lavrenko [Lavrenko 2004] used an expectation over the parameter space, described in more detail in the next section. In general, we can view different approximations to the scoring integral as different approaches to addressing the uncertainty encoded by the Bayesian formulation.

2.4.2 Monte Carlo integration in the GRM

We show how standard document scoring formulas in the GRM can be viewed as the result of two successive Monte Carlo-like approximations of different integrals.

First, consider the document likelihood ranking criterion for the GRM, as given by Eq. 2.21. As an example, we assume a unigram model, so that document $\boldsymbol{d} = d_0 d_1 \cdots d_n$, and

$$p_\theta(\boldsymbol{d}) = \prod_{i=1}^{n} p(d_i|\theta) \tag{2.23}$$

which leads to the ranking criterion

$$p(\boldsymbol{d}|\boldsymbol{q}) = \int_\Theta \prod_{i=1}^{n} p(d_i \mid \theta) dp_{\boldsymbol{q}}(\theta). \tag{2.24}$$

In his dissertation ([Lavrenko 2004], p.32), Lavrenko notes that the integral in Eq. 2.24 is computationally expensive and approximates it by evaluating the integrand at a single value $\hat{\theta}$ defined by the expected value over the parameter vector θ:

$$\hat{\theta} = \mathbb{E}_{\boldsymbol{q}}[\theta] = \int_\Theta \theta \cdot dp_{\boldsymbol{q}}(\theta) \tag{2.25}$$

which gives the document score

$$p(\boldsymbol{d}|\boldsymbol{q}) \approx \prod_{i=1}^{n} p(d_i|\hat{\theta}) = \prod_{i=1}^{n} p(d_i|\mathbb{E}_{\boldsymbol{q}}[\theta]) \tag{2.26}$$

To approximate non-relevant documents, we use the collection C to approximate $p(\boldsymbol{d}|C)$ in a similar way. This gives a final ranking criterion of

$$\frac{P(D = d \mid R = 1)}{P(D = d \mid R = 0)} \approx \frac{P_{\mathcal{R}}(D = d \mid \hat{\theta})}{P_{\mathcal{R}}(D = d \mid C)} = \prod_{i=1}^{n} \frac{p(d_i|\mathbb{E}_{\boldsymbol{q}}[\theta])}{p(d_i|\mathbb{E}_C[\theta])} \tag{2.27}$$

Viewing the measure $dp_q(\theta)$ essentially as a sampling density over Θ, it is evident that Eq. 2.27 is equivalent to a single-sample Monte Carlo estimate of Eq. 2.24 at the point $\hat{\theta}$. The expectation $\mathbb{E}_{\boldsymbol{q}}[\theta]$, however, is *itself* an integral over Θ that depends critically on how $dp_q(\theta)$ is defined. Therefore, we now show how the second Monte Carlo approximation, that of Eq. 2.25, is accomplished in Lavrenko's implementation of the GRM [Lavrenko

2004] by specifying $dp_q(\theta)$ using a method called *kernel-based density allocation*. The idea behind kernel-based density allocation is simple. Suppose we wish to create a density function over Θ that reflects a likely generating distribution for a set of m training examples $\theta_1, \cdots, \theta_m$. To do this, we define the density as a superposition of m simple *kernel* functions $k_i(\theta)$, each of which is centered on a single training point θ_i with local weight w_i. This gives

$$dp_q(\theta) = \sum_{i=1}^{m} w_i \cdot k_i(\theta) \tag{2.28}$$

One very simple kernel is the *Dirac* kernel which is given by

$$k_{0,1}^{\boldsymbol{q}}(\theta) = \begin{cases} 1 & \text{if } \theta = \theta_{\boldsymbol{q}} \\ 0 & \text{otherwise} \end{cases} \tag{2.29}$$

where $\theta_{\boldsymbol{q}}$ is the empirical distribution for a query $\boldsymbol{q}$. (Lavrenko also proposes, but does not evaluate, the use of a Dirichlet kernel ([Lavrenko 2004], p. 54–55).) We can think of the use of a Dirac kernel $k_i(\theta)$ as equivalent to a deterministic sampling distribution that has $p(\theta_i) = 1$, and $p(\theta) = 0$ everywhere else.

When we use the definition of Eq. 2.28 for $dp_q(\theta)$ with the integral in Eq. 2.25, the Dirac kernel formulation can be seen as equivalent to a multi-sample Monte Carlo approximation of $\mathbb{E}_{\boldsymbol{q}}[\theta]$ in which we take m samples over $dp_q(\theta)$ with values $\theta_0, \cdots, \theta_{m-1}$. Our contribution in Section 2.6 is to show that this multi-sample estimate is a special case of a sampling technique called *multiple importance sampling* that treats the m different kernel densities as m sampling strategies over Θ, obtains one sample from each strategy, and combines the samples using a provably good sample weighting.

The key point here is that the use of multiple samples with $m > 1$ represents the introduction of multiple hypotheses, as opposed to a single choice of model. Figure 2.1 shows a simple illustration of how different model choices, seen as samples, can be more or less effective at estimating a document's score for different queries. For some queries, the scoring integrand for relevant documents f_D tends to be largest in the neighborhood of the original query, and thus a single sample near the original query gives a better approximation than using the feedback model. For other topics, the opposite is true: relevant documents have highest scores near the feedback model, and thus a single feedback model sample gives the more reliable score. Combining these complimentary strategies amounts to finding a good

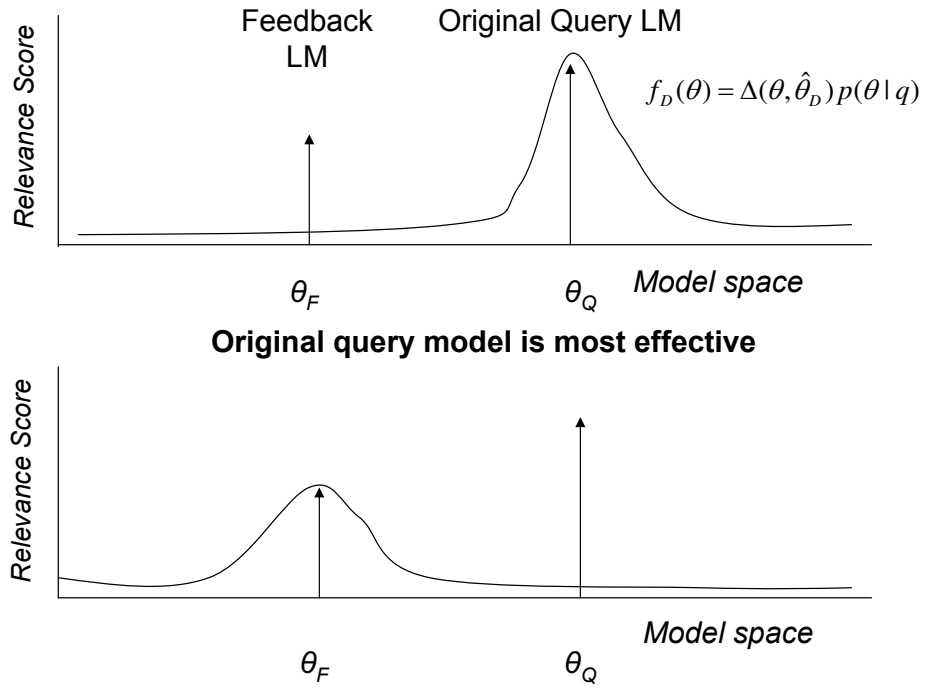


Figure 2.1: Model combination as a sample weighting problem, showing simplified view of a document score integrand f_D for two different query scenarios. Single-sample approximations using either an original query model (θ_Q) or a feedback model (θ_F) can give good or bad estimates of the score integral. In some cases, relevant documents have high scores for models near the original query (top). In other cases, relevance is better captured by a feedback model (bottom) that is far from the original query. Since we do not know the 'correct' choice in advance, we can manage risk by combining samples from multiple complementary strategies, thus 'hedging' our model choices and stabilizing the retrieval algorithm.

weighting scheme for the sample contributions. In the next section, we examine specific methods for choosing samples and calculating weights for them, focusing on importance sampling.

2.5 Sampling methods

In the previous section we saw how the standard document scoring formula for document-based ranking in the GRM was based on using a single-sample Monte Carlo approximation to an integral over the space of relevance model parameters. In this section we show how our Monte Carlo estimators may be extended when multiple samples and sampling strate-

gies are available.

One source of variance in the Monte Carlo estimate of an integral is a sampling distribution that poorly matches the shape of the integrand. Sampling is most effective when we sample in the parts of the domain where the integrand $f(x)$ is largest, or varies the most. The sampling methods described in this section attempt to reduce variance by getting a good 'fit' to the integrand.

We will not review the extensive literature on sampling methods here. Instead, we focus on those methods that are likely to best fit the types of problems we see in information retrieval.

2.5.1 Importance sampling

One widely-used technique for reducing variance is *importance sampling* ([Kalos & Whitlock 1986], p.92). The key principle of importance sampling is to use a sampling distribution p_i that is a close match for the shape of the integrand $f(\cdot)$. In particular, importance sampling works well when the integrand tends to have its largest values on limited areas in the domain. In the case of relevance functions, the neighborhood in parameter space around the original query model is of particular interest, for example.

We now describe a generalization of importance sampling that can use multiple sampling strategies for an integral and combine the samples in simple but effective ways.

2.5.2 Multiple importance sampling

Multiple importance sampling was introduced by Veach [Veach 1997] to deal with complex integrals for light transport in computer graphics. The key idea of multiple importance sampling is to combine several importance sampling techniques that make different assumptions about the nature of the integrand. In this way, the stability of the estimate is improved. In the information retrieval domain, the samples correspond to different models θ_i, and thus multiple importance sampling is a form of model combination. The sample (model) weights are given by various choices of heuristic, described below.

More formally, a *multi-sample Monte Carlo estimator* is a weighted combination of individual Monte Carlo estimators, each of which corresponds to one of n sampling tech-

niques $\mathcal{S}_i$ with sampling distributions $p_i(\cdot)$ and where n_i samples are taken from each $p_i(\cdot)$.

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \hat{w}_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{2.30}$$

There are different choices for choosing the weighting function $\hat{w}_i(X_{i,j})$ given to each sample $X_{i,j}$. One choice is simply to set fixed weights. Unfortunately, if any particular sampling technique has high variance for a particular problem, F itself will also have high variance. However, we can considerably improve on fixed weighting by using the following family of heuristics defined by Veach [Veach 1997]. Proofs of the various properties of these estimators we state here are given in Chapter 10 of Veach's thesis.

The balance heuristic

The following weighting function $\hat{w}(x)$, termed the *balance heuristic*, happens to have a very simple form, defined by the weighted combination

$$\hat{w}_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)} \tag{2.31}$$

Here, x is a sample (model) and n_i is the number of samples (models) taken using the i-th sampling strategy. The balance heuristic has the property that the variance of F can be shown to never be much worse that the variance of any other weighted linear combination ([Veach 1997], p. 264).

Note that the balance heuristic can be reformulated as regular importance sampling over a combined sampling distribution

$$\hat{p}(x) = 1/n \sum_{i=1}^{n} p_i(x) \tag{2.32}$$

The requirements for using a sampling technique with the balance heuristic are only slightly greater than the minimal Monte Carlo requirements of sample generation and point evaluation. Given n sampling techniques and a sample $X_{i,j}$ generated from sampling distribution $p_i(x)$, we must be able to evaluate the probability $p_k(X_{i,j})$ that the *other* $n - 1$ sampling techniques generate $X_{i,j}$.

The power heuristic

A generalization of the balance heuristic is the *power heuristic*, which raises the weights to an exponent β. The intent of the power heuristic is to sharpen the weighting functions and reduce variance in cases where one of the score sampling distributions is a good fit for the term score integrand.

$$\hat{w}_i(x) = \frac{(n_i p_i(x))^\beta}{\sum_k (n_k p_k(x))^\beta} \tag{2.33}$$

As β is increased, large weights move closer to one and small weights move closer to zero. In the limit, we obtain the *maximum heuristic* which ignores all $p_i(\cdot)$ except the largest one. In practice, however, the maximum heuristic is not as effective because it discards too much evidence.

2.5.3 Stratified sampling

One traditional variance-reduction method is *stratified sampling* ([Kalos & Whitlock 1986], p.112), which partitions the domain Θ into n non-overlapping regions Θ_i and takes n_i samples from each region according to a density function p_i.

If Θ is the space of unigram language models, a stratum Θ_i might correspond to a hypercube in which the probability of each word $\theta_{i,w}$ lies in a pre-defined range $[\alpha_{i,w}, \beta_{i,w}]$. If we associated language models with queries, we could use stratified sampling to generate queries that were evenly sampled among the possibly interesting combinations of query terms, where the strata were defined using thresholds on term probabilities.

Stratified sampling works well (e.g. has good convergence properties) when the dimension of the domain is low and the integrand is well-behaved. ([Veach 1997] p.50). The benefits of stratified sampling can be diminished for information retrieval problems: the relevance function integrand is potentially complex, and for performance reasons the number of samples (queries) that we have to work with is usually very small. In addition, the dimensionality – as determined by vocabulary size – would also need to be restricted to a relatively small subset.

2.5.4 One-sample models

A different sampling method, called *one-sample* estimation, operates by first randomly selecting a sampling method S_i according to the distribution p_s, and then taking a single sample according to the corresponding sampling distribution p_i. In this case, the balance

heuristic can be shown to be optimal [Veach 1997]. In a sampling view of query expansion, one-sample models would amount to selecting one of the top-retrieved documents according to $p(d|q)$ and then selecting a query from the terms in the document. The power heuristic above corresponds to raising all document retrieval probabilities to a power β and then normalizing them[3].

2.5.5 Deterministic sampling

Because information retrieval evaluation must be extremely fast, the number of samples we can apply with Monte-Carlo-type methods is typically small (less than 10). If we make assumptions about the distribution of, say θ in Θ, then we can explore deterministic methods for sampling.

One technique for computing statistics of non-linear functions of a random variable is the *unscented transform* [Julier & Uhlmann 2002]. Suppose $f(\theta)$ is a scoring function, and suppose we know the density $h(\theta)$ of θ is distributed as a d-dimensional Gaussian with mean μ and covariance matrix Σ. Then to approximate the expectation $\int f(\theta)h(\theta)$ we choose $2d$ points x_k for $k = 1, \ldots, 2d$ such that

$$x_k = \mu + (\sqrt{d\Sigma})_k \tag{2.34}$$

$$x_{d+k} = \mu - (\sqrt{d\Sigma})_k \tag{2.35}$$

where $(\sqrt{d\Sigma})_k$ is the k-th column of the matrix square root of Σ. The matrix square root is defined such that if UDU^T is the singular value decomposition of Σ, with $U = \{U_1, \ldots, U_d\}$ and $D = diag\{\lambda_1, \ldots, \lambda_d\}$ then $(\sqrt{d\Sigma})_k = \sqrt{\lambda_k}U_k$. The sample points x_k effectively summarize the mean and variance of $h(\theta)$ and are then used in the following Monte-Carlo-like approximation:

$$\int f(x)h(x)dx \approx \frac{1}{2d}\sum_{k=1}^{2d} h(x_k). \tag{2.36}$$

This method can be generalized to include μ and scaled versions of x_k as additional sample points. We call the use of the unscented transform in this way *sigma-point sampling*.

The idea of using sigma points to replace the single-point estimate of Eq. 2.25 (re-

[3]This is reminiscent of a popular heuristic in natural language processing and elsewhere that squares probability estimates and then normalizes them to obtain relative hypothesis weights, instead of using the initial probability estimates.

placing the Gaussian assumption in the example above with a Dirichlet assumption on θ) is appealing because it allows us to replace Dirac functions with effective approximations of more realistic kernel functions that would otherwise be too computationally intensive to use. Sigma-point sampling is also useful in constructing similarity measures for retrieval, as we discuss further in Chapter 4.

2.5.6 Closed form solutions

In certain special cases, an integral may have a closed-form exact solution, making sampling-based or other approximations unnecessary. For example, as noted by [Dillon et al. 2007] in their recent work on text classification, if we have a document unigram model for document D with parameters θ_D, a query unigram model for query Q with parameters θ_Q, and a word-word translation model matrix $T_{ij} = p(t_i|t_j)$ the expected quadratic distance between the two models with respect to the distribution induced by T can be written in closed form after some algebra. The initial integral is

$$d(\theta_Q, \theta_D) = \int ||\theta_{\bar{Q}} - \theta_{\bar{D}}||_2^2 p(\theta_{\bar{Q}}|\theta_Q) p(\theta_{\bar{D}}|\theta_D) d\theta_{\bar{Q}} d\theta_{\bar{D}} \tag{2.37}$$

and this can be rewritten as

$$d(\theta_Q, \theta_D) = \int \langle \theta_{\bar{Q}}, \theta_{\bar{Q}} \rangle p(\theta_{\bar{Q}}|\theta_Q) d\theta_{\bar{Q}} \quad + \int \langle \theta_{\bar{D}}, \theta_{\bar{D}} \rangle p(\theta_{\bar{D}}|\theta_D) d\theta_{\bar{D}} \tag{2.38}$$

$$- 2 \int \langle \theta_{\bar{Q}}, \theta_{\bar{D}} \rangle p(\theta_{\bar{Q}}|\theta_Q) p(\theta_{\bar{D}}|\theta_D) d\theta_{\bar{Q}} d\theta_{\bar{D}} \tag{2.39}$$

resulting in the closed form solution

$$d(\theta_Q, \theta_D) = N_1^{-2} \sum_{i=1}^{N_1} \sum_{j \in \{1,\ldots,N_1\} \setminus \{i\}} (TT^\top)_{q_i, q_j} + N_2^{-2} \sum_{i=1}^{N_2} \sum_{j \in \{1,\ldots,N_2\} \setminus \{i\}} (TT^\top)_{d_i, d_j} \tag{2.40}$$

$$- 2N_1^{-1} N_2^{-2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (TT^\top)_{q_i, d_j} + N_1^{-1} + N_2^{-1} \tag{2.41}$$

where N_1 and N_2 are the length in words of the query and document respectively, q_i is the vocabulary index for the i-th word of Q, d_i is the vocabulary index for the i-th word of D, and T is the word-word matrix of translation probabilities.

While scoring functions in information retrieval can take a similar form, useful distance

functions between documents and queries may be much more complex than squared loss. The Indri query language [Strohman et al. 2004], for example, allows passages and documents to be scored using arbitrarily complex functions that are the output of inference networks described via a structured query language. We therefore want to choose methods that can handle these more complex loss functions and in general make few assumptions about the integrand, and so while closed-form solutions may prove very useful for some problems, we emphasize much more general approximations method like importance sampling in most of our work.

2.6 Document-based ranking as a special case of the balance heuristic

As an example of how a sampling-based view of information retrieval can lead to new insight into existing methods, we now show that the standard formula for document-based ranking in the GRM is actually a special case of the balance heuristic, using one sampling strategy for each top-retrieved document.

Theorem 1. *Given a collection of N documents $\mathcal{D} = \{d_1, \ldots, d_i, \ldots, d_N\}$, a query $\boldsymbol{q}$, and a document $\boldsymbol{d} \in \mathcal{D}$, let $\theta_{\boldsymbol{d}}$ and $\theta_{\boldsymbol{q}}$ be unigram models (say) in the relevance model space Θ corresponding to $\boldsymbol{d}$ and $\boldsymbol{q}$ respectively. Let $p_i(\theta) = p(\theta \mid d_i)$ for each document $d_i \in \mathcal{D}$. Then the GRM document-based ranking formula*

$$R_D(\boldsymbol{d}, \boldsymbol{q}) = \frac{\sum_{i=1}^{N} p_i(\theta_{\boldsymbol{d}}) p_i(\theta_{\boldsymbol{q}})}{\sum_{i=1}^{N} p_i(\theta_{\boldsymbol{q}})}$$

is a special case of the balance heuristic.

Proof. We can write the document scoring function as an integral over the relevance model parameter space Θ:

$$R_D(\boldsymbol{d}, \boldsymbol{q}) = p(\boldsymbol{d} \mid \boldsymbol{q}) = \int_{\Theta} p(\boldsymbol{d} \mid \theta) p(\theta \mid \boldsymbol{q}) d\theta \tag{2.42}$$

For each document $d_i \in \mathcal{D}$ we define a sampling distribution $p_i(\theta) = p(\theta \mid d_i)$.[4] A Monte Carlo estimator for R_D using samples $X_{i,j} \in \Theta$ is therefore

$$\hat{R}_D(d,q) = \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \hat{w}_i(X_{i,j}) \frac{p(\boldsymbol{d} \mid X_{i,j}) p(X_{i,j} \mid \boldsymbol{q})}{p_i(X_{i,j})} \tag{2.43}$$

where n_i is the number of samples from distribution p_i. Setting $\hat{w}$ according to the balance heuristic, we obtain

$$\hat{R}_D(d,q) = \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{p(\boldsymbol{d} \mid X_{i,j}) p(X_{i,j} \mid \boldsymbol{q})}{\sum_{k=1}^{N} p_k(X_{i,j})}$$

We view $\theta_{\boldsymbol{q}}$ as a single sample from the true relevance distribution defined on Θ, and for each p_i we let $X_{i,j} = \theta_{\boldsymbol{q}}$ and $n_i = 1$. Since $p(\theta_{\boldsymbol{q}} \mid q)$ is the same for all documents, this gives

$$\hat{R}_D(d,q) \overset{rank}{=} \frac{1}{N} \sum_{i=1}^{N} \frac{p(\boldsymbol{d} \mid \theta_{\boldsymbol{q}})}{\sum_{k=1}^{N} p_k(\theta_{\boldsymbol{q}})}$$

If we denote d_m as any document *not* in the collection $\mathcal{D}$, we assume that d_m cannot be retrieved by the query and thus $p(d_m \mid \theta_{\boldsymbol{q}}) = 0$. This implies that

$$p(\boldsymbol{d} \mid \theta_{\boldsymbol{q}}) = \sum_{k=1}^{N} p(\boldsymbol{d} \mid d_k) p(d_k \mid \theta_{\boldsymbol{q}})$$

so that

$$\begin{aligned} \hat{R}_D(d,q) &\overset{rank}{=} \sum_{i=1}^{N} \frac{\sum_k p(\theta_{\boldsymbol{d}} \mid d_k) p(d_k \mid \theta_{\boldsymbol{q}})}{\sum_k p_k(\theta_{\boldsymbol{q}})} \\ &\overset{rank}{=} \frac{\sum_{i=1}^{N} p_i(\theta_{\boldsymbol{d}}) p_i(\theta_{\boldsymbol{q}})}{\sum_{i=1}^{N} p_i(\theta_{\boldsymbol{q}})} \end{aligned} \tag{2.44}$$

which is the desired GRM document ranking formula. □

In retrospect, we can see at least one reason why the GRM formula may be effective: since at least a few documents in the top retrieved set are likely to be relevant, at least some

[4] For a practical implementation we would select a top-ranked subset of documents instead of the entire collection.

of the sampling strategies will be a good fit for the shape of the true relevance distribution.

Suppose that instead of a single query from the user, we observe a set of queries $\{q_i\}$ that are different attempts to express the same information need. This important scenario has been studied extensively in IR, especially with respect to user modelling. Previous influential work was the I^3R project of Croft and Thompson [Croft & Thompson 1987] in the late 1980s, which used data fusion techniques to obtain improved results from combining multiple representations. The I^3R project helped shape the design of TREC's current topic format, in which three successively more detailed expressions of the same information need are given: the 'title', 'description', and 'narrative' fields.

When considering how to combine these types of multiple query representations, the balance heuristic gives us a simple way to generalize Eq. 2.44 to include this information.

Theorem 2. *Given a set $\mathcal{Q}$ of m queries with models θ_j sampled from a distribution $\mathcal{P}_\mathcal{I}$ for the same information need $\mathcal{I}$, the document ranking formula in Eq. 2.44 generalizes to*

$$R_D(\boldsymbol{d}, \mathcal{Q}) = \sum_{i=1}^{N} \sum_{j=1}^{m} m \cdot \frac{p_i(\theta_j)}{\sum_d p_i(\theta_j)} p_i(\boldsymbol{d}) \tag{2.45}$$

In addition to single-user scenarios, one application of a result like Eq. 2.45 would be to use query logs from Web search engines to identify likely variations of the same query from different users, which could then be used as samples in the above formula.

2.7 Other examples of sampling in IR

Other uses of sampling have recently made their way to IR. Bar-Yossef and Gurevich [Bar-Yossef & Gurevich 2006] evaluate a number of sampling techniques, including the Metropolis algorithm, to obtain near-uniform samples from a Web search engine's index. Anagnostopoulos [Anagnostopoulos et al. 2005] et al. use random sampling of search results to correct for bias in various applications that results from only considering the top k documents. They note that these applications include estimating the number of relevant documents, finding terms associated with the query terms, and clustering top results that give a more complete covering of all aspects of the query results. The authors focus on efficient implementation of random result sampling in the search engine itself.

2.8 Summary

The primary contribution of this chapter is to introduce the use of sampling to compute important quantities such as query variants, document scores, variances, and approximation posterior distributions in information retrieval. Along the way, we also defined some basic concepts in probability and statistics, such as expectation and variance, that will prove useful in later chapters. A key motivation for sampling is the idea that queries and documents are noisy observations or translations from some ideal latent query or document model space. As such, they should be treated as *random variables* and not single fixed observations. This leads to the idea that entities such as scoring functions become *integrals* over the distributions of the random variables. This in turn leads to the introduction of sampling methods known as *Monte Carlo integration* methods for approximating these potentially complex integrals. We discussed several approaches to sampling, especially *multiple importance sampling* and heuristics for sample weighting.

Powerful Bayesian formulations such as the Generative Relevance Model [Lavrenko 2004] and Risk Minimization framework [Zhai & Lafferty 2006] are theoretical models that incorporate the idea of queries and documents as random variables. However, in practice, implementations of these models have failed to exploit their ability to account for multiple potential query 'translations' or document expansions, each of which can be seen as the result of some choice of translation process – and thus, sampling strategy – on the integral. In Chapters 3, 4 and 6 we show that the choice of sampling strategy can have a significant impact on task performance. Thus, finding sampling strategies that are reliable and effective is a new research question – one that we begin to explore in this thesis.

As a specific example of the insights gained from a sampling-based view of retrieval scoring, we showed how simple deterministic sampling methods are examples of efficient approximations the document scoring integrals that arise in the Generative Relevance Model (GRM). We proved that the unigram GRM document scoring function can be seen as a special case of multiple importance sampling, with one sampling strategy per top-ranked document, and the balance heuristic used for the weighted model combination.

We chose the GRM for analysis because of its simple (but significant) assumption that a single underlying latent joint relevance distribution generates both documents and queries. However, it is important to note that the connection between Monte Carlo integration methods can be applied to any statistical approaches to retrieval in which it makes sense to in-

tegrate over a space of models, including the Bayesian formulations mentioned above. In Chapter 3, we follow the practical implications of treating documents and queries as random variables by forming a new general framework for improved query model estimation.

Chapter 3

A Theoretical Framework for Robust Pseudo-Relevance Feedback

In Chapter 1 we discussed *pseudo-relevance feedback*, a method of automatic query expansion that attempts to improve retrieval performance by enhancing the query with terms from the first k top-ranked documents, which are assumed relevant. In this chapter we explore how sampling can be used to improve the performance of pseudo-relevance feedback. The key idea is that existing pseudo-relevance feedback methods typically perform averaging over the top-retrieved documents, but ignore an important statistical dimension: the risk or variance associated with the underlying retrieved document sets and their relevance weights from which the feedback model is calculated. Intuitively, by using sampling, we can smooth out this risk over several models, to obtain a combined model with more consistent performance.

We propose a general retrieval framework in which we define sampling distributions over important entities such as the query and top-retrieved documents. The samples are then used as input to a baseline feedback algorithm. We show how sampling can be used to obtain estimates of algorithm variance and sensitivity, which in turn can be used to improve retrieval quality in various ways. For example, in the case of pseudo-relevance feedback we find that sampling top documents helps increase individual feedback model precision by reducing noise terms, while sampling from the query improves robustness (worst-case performance) by emphasizing terms related to multiple query aspects. The result is a meta-feedback algorithm that is both more robust and more precise than the

original strong baseline method.

Our main goal in this chapter is to use estimates of parameter uncertainty to improve retrieval quality, especially the quality of feedback models, as evaluated by important measures such as precision and robustness (worst-case performance). Our basic strategy will be to first generate multiple estimators for a given task, such as finding a good feedback model. Each estimator represents a different strategy or set of assumptions about how a good model would be derived. By using multiple estimators/strategies, we can essentially hedge our bets so that we are not committing completely to a single strategy, while preserving significant gains in case one strategy turns out to be very effective. These multiple estimators are then combined in a Bayesian framework in a way that accounts for the uncertainty or confidence in the component models. In this way, we aim to both improve the accuracy of the final estimator, and increase stability in the predictions.

This chapter is organized as follows. Section 3.1 describes the general retrieval framework we use to perform enhanced feedback. We discuss specific methods for model combination in Section 3.2. Because the study of robustness is quite new, we introduce new evaluation methods for it in Section 3.3, including *risk-reward curves* which are a very important summary of performance throughout this thesis. The evaluation is contained in Section 3.4, and related work is covered in Section 3.5. We analyze the computational costs of the framework in Section 3.6. Section 3.7 describes a few possible future extensions. We conclude by summarizing the key contributions of our approach in Section 3.8.

3.1 General retrieval framework

We now give a description of how a retrieval framework can be defined based on sampling methods. We define some basics, and then describe our sampling framework in detail.

3.1.1 Basic concepts and notation

We assume a user $\mathcal{U}$ has an information need Q that is expressed as a specific query $\boldsymbol{q}$ to the search engine. Typically $\boldsymbol{q}$ consists of a set of search terms $\boldsymbol{q} = q_1 \dots q_N$. We assume a retrieval system that processes the query $\boldsymbol{q}$ by assigning a real-valued document score $f(\boldsymbol{d}, \boldsymbol{q})$ to each document $\boldsymbol{d}$ in the collection C, where $C = \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_C\}$. We also make the very general assumption that the scores reflect some degree of relevance, and apply the *Probability Ranking Principle* [Robertson 1977], which makes the following two assumptions, as stated by Robertson:

- The *relevance* of a document to a request is independent of the other documents in the collection.
- The *usefulness* of a relevant document to a requester may depend on the *number* of relevant documents the requester has already seen (the more he has seen, the less useful a subsequent one may be).

With these assumptions, we are justified in ranking documents by decreasing score $f(\boldsymbol{d}, \boldsymbol{q})$ which can be calculated independently for each document. Essentially, this assumption also implies that we can use the score to assign reasonable pseudo-relevance weights to documents: either directly, if the retrieval model gives document scores proportional to the estimated probability of relevance, or indirectly according to rank otherwise. We make no other assumptions about $f(\boldsymbol{d}, \boldsymbol{q})$.

The nature of $f(\boldsymbol{d}, \boldsymbol{q})$ may be complex. for example, if the retrieval system supports structured query languages [Strohman et al. 2004], then $f(\boldsymbol{d}, \boldsymbol{q})$ may represent the output of an arbitrarily complex inference network defined by the structured query operators. In theory, this scoring function can also vary from query to query, although in this work for simplicity we keep the scoring function the same for all queries.

As a basis to represent and compare documents and queries we use the language modeling (LM) approach for information retrieval [Ponte & Croft 1998]. In this view, a text T is treated as a sequence of terms $t_1 \ldots t_N$ that was generated by a statistical model with parameters θ_T. For simplicity, we use a unigram language model, which is basically a word histogram. A unigram model assumes that each word t_i is generated independently of the other words. We refer to the probability of term w in language model $\hat{\theta}_{\boldsymbol{q}}$ by $\hat{\theta}_{\boldsymbol{q}}[w]$. More sophisticated language models are possible that capture more structure in text, such as word order or topic structure. For now, however, we start by using unigram models.

Let $\hat{\theta}_{\boldsymbol{q}}$ and $\hat{\theta}_{\boldsymbol{d}}$ denote the parameters of language models estimated for a query and document respectively. The model $\hat{\theta}_{\boldsymbol{q}}$ is selected by user $\mathcal{U}$ according to the distribution $p(\hat{\theta}_{\boldsymbol{q}}|\mathcal{U}, Q)$. The actual observed query $\boldsymbol{q}$ is considered to be generated with probability $p(\boldsymbol{q}|\hat{\theta}_{\boldsymbol{q}})$.

A scoring function f in the language modeling approach can take many different forms. One widely-used function to compare two probability distributions is the *KL-divergence*[1] measure $f(\theta_1, \theta_2) = KL(\theta_1||\theta_2)$.

[1] KL-divergence is defined in Appendix A,Section A.1

Whatever the specific algorithm, we denote the set of k retrieved documents for $\boldsymbol{q}$ having the largest values of f in collection C by $\mathcal{D}_k(\boldsymbol{q}, f, C)$. For clarity we assume that f, k, and C are fixed and just write $\mathcal{D}$ when the query is understood, or $\mathcal{D}_{\boldsymbol{q}}$ for an explicit query $\boldsymbol{q}$. Each document $\boldsymbol{d}_i$ in $\mathcal{D}$ has a corresponding score $w_{\boldsymbol{d}_i} = f(\boldsymbol{d}_i, \boldsymbol{q})$.

3.1.2 Pseudo-relevance feedback

Once we have the set $\mathcal{D}$, we can perform *pseudo-relevance feedback* (PRF). We treat a pseudo-relevance feedback algorithm as a black box function $\Phi(\boldsymbol{q}, \mathcal{D})$ whose input is a query $\boldsymbol{q}$ and top-retrieved document set $\mathcal{D}$ and whose output is a feedback language model θ_F. [2]

To incorporate feedback in the LM approach, we assume a model-based scheme in which our goal is to take the query and resulting ranked documents $\mathcal{D}$ as input, and estimate a feedback language model $\hat{\theta}_F$, which is then interpolated with the estimated original query model $\hat{\theta}_{\boldsymbol{q}}$:

$$\hat{\theta}_{New} = (1 - \alpha) \cdot \hat{\theta}_{\boldsymbol{q}} + \alpha \cdot \hat{\theta}_F \tag{3.1}$$

This includes the possibility of $\alpha = 1$ where the original query mode is completely replaced by the feedback model. In the next section, we explain how sampling can be applied to pseudo-relevance feedback and the calculation of $\hat{\theta}_F$.

3.1.3 A resampling approach to pseudo-relevance feedback

Instead of running Φ once using the observed $\boldsymbol{q}$ and $\mathcal{D}$ to get a single $\hat{\theta}_F$, we treat the inputs to the feedback black box as random variables. We are interested in quantifying the uncertainty of the feedback model $\hat{\theta}_F$ as a random variable that changes in response to small changes in $\boldsymbol{q}$ and $\mathcal{D}$. We will then use this knowledge to combine multiple feedback models to produce a more robust final query model.

Like the document scoring function $f(\boldsymbol{d}, \boldsymbol{q})$, the feedback algorithm $\Phi(\boldsymbol{q}, \mathcal{D})$ may implement a complex, non-linear scoring formula, and so as $\boldsymbol{q}$ and $\mathcal{D}$ are varied, the resulting feedback models may have a complex distribution over the space of feedback models (the *sample space*). We denote the unknown density of θ_F by $p(\theta|\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is a vector of parameters. Note that $p(\theta|\boldsymbol{\alpha})$ is a distribution over language models.

We would like to build up a picture of approximately what $p(\theta|\boldsymbol{\alpha})$ looks like as the

[2] Other factors, such as the collection C are also part of the implicit input, but for clarity we omit these from the notation.

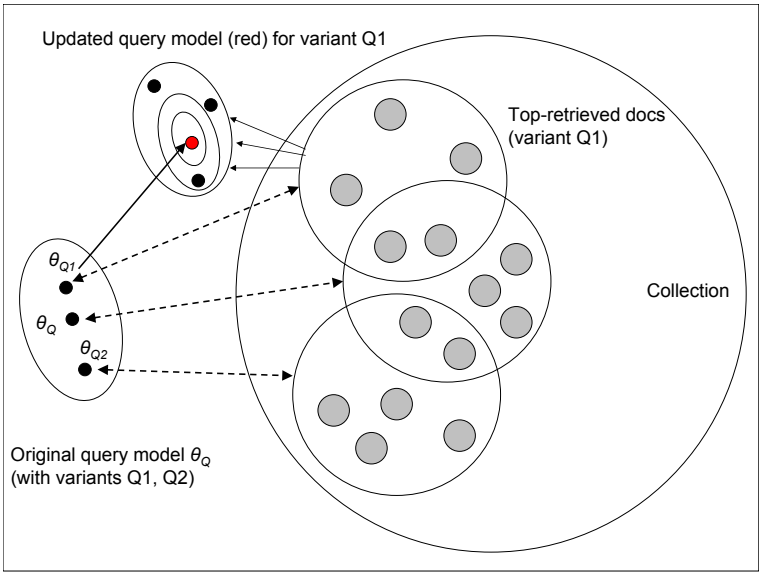


Figure 3.1: General retrieval framework that treats queries and top-retrieved document sets as random variables. Samples are taken in the form of variations on the original query and documents.

baseline feedback method Φ is run many times with slightly different inputs, To accomplish this, we create a small number of query variants $\boldsymbol{q}_i$ from the initial query $\boldsymbol{q}$: a process we call *query resampling*. For each $\boldsymbol{q}_i$, we then perform *document set resampling* by creating a small number of variants of the top-ranked document set returned for $\boldsymbol{q}_i$. The entire process is shown in Figure 3.1. We now describe document set resampling first in Section 3.1.4 and then query resampling in Section 3.1.5.

3.1.4 Document set resampling

To create variants of the top-ranked document set for a given query, we apply a widely-used simulation technique called *bootstrap sampling* ([Duda et al. 2001], p. 474). Bootstrap sampling allows us to simulate the approximate effect of perturbing the parameters within the black box feedback algorithm by perturbing its inputs in a systematic way, while making few assumptions about the nature of the feedback algorithm Φ.

More specifically, we randomly select k documents *with replacement* from $\mathcal{D}$. This sampling can also be deterministic, and there is a sizable literature on enumerating 'good' subsets using Gray codes and other methods [Diaconis & Holmes 1994]. Whatever sampling method is used, the result is a new document set $\mathcal{D}_{(i)}$. With this set, we then calculate

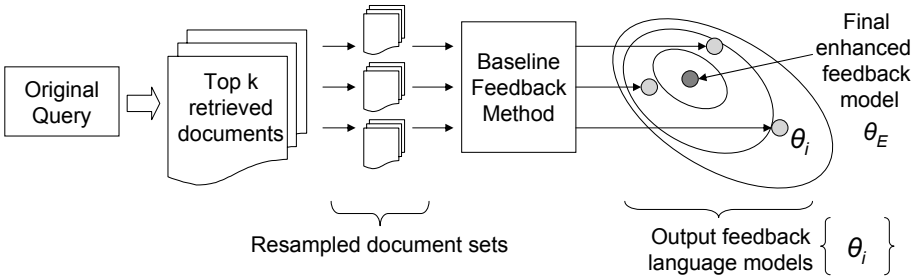
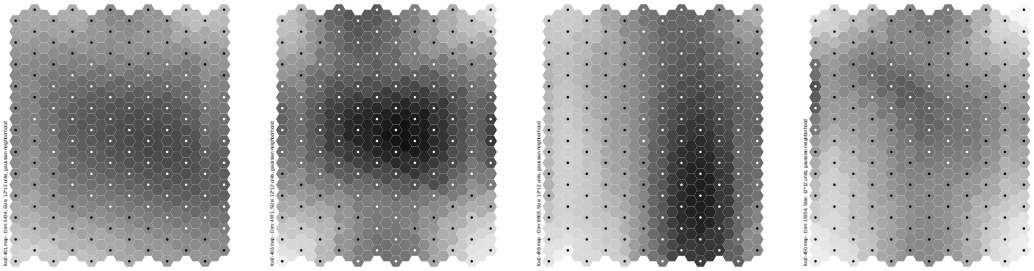


Figure 3.2: How bootstrap sampling over the initial top-ranked document set is used to create an output distribution over the sample space of possible feedback models.

a modified feedback language model $\theta_{(i)} = \Phi(\boldsymbol{q}, \mathcal{D}_{(i)})$. We repeat this process B times to obtain a set of B feedback language models, which we denote $\{\boldsymbol{\theta_{(q,i)}}\}$ and which will be used to fit a distribution $p(\theta|\boldsymbol{\alpha})$. We will then obtain a final feedback model $\hat{\theta}_F^{\boldsymbol{q}}$ by choosing a representative model from $p(\theta|\boldsymbol{\alpha})$, such as the mode or mean. Typically B is in the range of 20 to 50 samples, with performance being relatively stable in this range. Figure 3.2 visualizes this process.

In traditional bootstrap sampling, each element of the training set (top-ranked documents) is equally likely to be chosen. Instead of treating each $\boldsymbol{d}$ in $\mathcal{D}$ as equally likely, however, an alternative is to weight the likelihood that a particular document will be chosen by its estimated probability of relevance, given either by its document score $w_{\boldsymbol{d}_i}$, or some decreasing function of $\boldsymbol{d}_i$'s rank. In this way, a document is more likely to be chosen the higher it is in the ranking. This may be desirable because it may reduce noise by focusing on fewer higher-quality documents – while carrying the risk of shrinking the coverage of important relevant concepts from lower-ranked documents.

We can model this weighted selection of top documents by using a simple form of *parametric bootstrap*. In the parametric bootstrap, we estimate the parameters of some distribution to fit the data, and then take random samples from that distribution. Here, we can consider a multinomial distribution over topics, treating each document as its own topic, and using the document's query likelihood score as its selection probability in the multinomial. A set of documents of given size is then considered as a random draw from this multinomial. In addition to uniform and multinomial distributions, other sampling strategies are also possible. For example, we could model variation in the relative proportions of the scores $w_{\boldsymbol{d}}$ using a Dirichlet distribution. We evaluate different sampling schemes for $\mathcal{D}$ in Section 3.4.9.



(a) Topic 401
Foreign minorities, Germany

(b) Topic 403
osteoporosis

(c) Topic 459
When can a lender foreclose on property

(d) Topic 460
Who was Moses

Figure 3.3: Visualization of expansion language model variability using self-organizing maps, showing the distribution of language models that results from resampling the inputs to the baseline expansion method. Dark areas represent regions of high model density. The similarity function is Jensen-Shannon divergence. The language model that would have been chosen by the baseline expansion is at the center of each map. Note that for some queries, such as topic 459 (Fig. 3.3c), the mode of the resampled distribution (in the darkest area of the map) differs significantly from the baseline expansion choice (at the center of each map).

Visualizing feedback distributions

Before describing how we use $\{\boldsymbol{\theta}_{(\boldsymbol{q},\boldsymbol{i})}\}$ to obtain a combined model $\hat{\theta}_F^{\boldsymbol{q}}$, it is instructive to view some examples of actual feedback model distributions that result from bootstrap sampling the top-retrieved documents from different TREC topics.

Each point in our sample space is a language model, which typically has several thousand dimensions. To help analyze the behavior of our method we used a Self-Organizing Map (via the SOM-PAK package [Kohonen et al. 1996]), to 'flatten' and visualize the high-dimensional density function[3].

The density maps for several TREC topics are shown in Figure 3.3 above. The *dark* areas represent regions of high similarity between language models. The *light* areas represent regions of low similarity – the 'valleys' between clusters. Each diagram is plotted so that the language model that would have been chosen by the baseline expansion is at the *center* of each plot. A single peak (mode) is evident in examples such as Fig.3.3b, but more complex structure appears in others, as in Fig.3.3d. Also, while the peak is often

[3]Because our points are language models in the multinomial simplex, we extended SOM-PAK to support Jensen-Shannon divergence, a widely-used similarity measure between probability distributions. Jensen-Shannon divergence and related measures are defined in Appendix A, Section A.1

close to the baseline feedback model, for some topics they are a significant distance apart (as measured by Jensen-Shannon divergence), as in Fig. 3.3c. In such cases, the mode or mean of the feedback distribution often performs significantly better than the baseline (and in a smaller proportion of cases, significantly worse).

Fitting a feedback model distribution

Because the feedback sample space is potentially complex, we do not attempt to derive a posterior distribution in closed form, but instead use simulation. We call the density $p(\theta|\boldsymbol{\alpha})$ over possible feedback models the *feedback model distribution*, where $\boldsymbol{\alpha}$ is a set of parameters for the distribution. Our goal in this section is to fit a useful parametric distribution $p(\theta|\boldsymbol{\alpha})$ using the sampled feedback models $\{\boldsymbol{\theta_{(q,i)}}\}$. We then select a representative model from $p(\theta|\boldsymbol{\alpha})$, such as the mode $\hat{\theta}_F^{\boldsymbol{q}}$, as the final expansion model for $\boldsymbol{q}$.

Our sample space is the set of all possible language models $\mathcal{L}_F$ that may be output as feedback models. Our approach is to take samples from this space and then fit a distribution to the samples using maximum likelihood. For simplicity, we start by assuming the latent feedback distribution has the form of a Dirichlet distribution. Although the Dirichlet is a unimodal distribution, and in general quite limited in its expressiveness in the sample space, it is a natural match for the multinomial language model, can be estimated quickly, and can capture the most salient features of confident and uncertain feedback models, such as the overall spread of the distibution.

After obtaining feedback model samples by resampling the feedback model inputs, we estimate the feedback distribution. We assume that the multinomial feedback models $\{\hat{\theta}_1, \ldots, \hat{\theta}_B\}$ were generated by a latent Dirichlet distribution with parameters $\{\alpha_1, \ldots, \alpha_N\}$. To estimate the $\{\alpha_1, \ldots, \alpha_N\}$, we fit the Dirichlet parameters to the B language model samples according to maximum likelihood using a generalized Newton procedure, details of which are given in Minka [Minka 2000b]. We assume a simple Dirichlet prior over the $\{\alpha_1, \ldots, \alpha_N\}$, setting each to $\alpha_i = \mu \cdot p(w_i \mid C)$, where μ is a parameter and $p(\cdot \mid C)$ is the collection language model estimated from a set of documents from collection C. The parameter fitting converges very quickly – typically just 2 or 3 iterations are enough – so that it is practical to apply at query-time when computational overhead must be small. A further approximation for speed is to restrict the calculation to a subset $\mathcal{V}_{\mathcal{D}}$ of the collection vocabulary $\mathcal{V}$ using the first k top-ranked documents (e.g. $k = 1000$), since we assume $\mathcal{V}_{\mathcal{D}}$ covers the great majority of relevant terms. Note that for this step we are re-using the

existing retrieved documents and not performing additional queries.

Given the parameters of an N-dimensional Dirichlet distribution $Dir(\alpha)$ the mean μ and mode x vectors are easy to calculate and are given respectively by

$$\mu_i = \frac{\alpha_i}{\sum \alpha_i} \quad (3.2) \quad \text{and} \quad \mathrm{x}_i = \frac{\alpha_i - 1}{\sum \alpha_i - N}. \quad (3.3)$$

We can then choose the combined model $\hat{\theta}_F^{\boldsymbol{q}}$ as mean or the mode of $p(\theta|\boldsymbol{\alpha})$ as the final enhanced feedback model. In practice, we found the mode to give slightly better performance.

Document set sampling strategies

The top-retrieved documents from the original query represent an important source of evidence for relevance. If estimated probabilities of relevance for each document are available, this creates a set of relative weights over the documents. The following two methods were tested in our evaluation.

- *Uniform selection* This strategy ignores the relevance scores from the initial retrieval and gives each document in the top k the same probability of selection in a bootstrap sample.

- *Rank-weighted selection* This strategy chooses documents with probability proportional to their relevance scores, if available, or a rank-based weighting otherwise. In this way, documents that were more highly ranked are more likely to be selected. When relevance scores are available, the observed weights are the relative scores. Otherwise, the reciprocal of the rank position may be used as a weight.

3.1.5 Query resampling

In the previous section, we calculated a combined feedback model $\hat{\theta}_F^{\boldsymbol{q}}$ from a set of resampled feedback language models $\{\boldsymbol{\theta_{(q,i)}}\}$ using document sets $\{\mathcal{D}_i\}$ as input to a baseline feedback method $\Phi(\boldsymbol{q}, \{\mathcal{D}_i\})$, while holding $\boldsymbol{q}$ fixed. The $\{\mathcal{D}_i\}$ were obtained by sampling from $\mathcal{D}_{\boldsymbol{q}}$ using a sampling distribution $p_{\mathcal{D}}(\cdot)$.

Instead of keeping $\boldsymbol{q}$ fixed, we can consider also defining a query sampling method that generates variants $\boldsymbol{q}_i$ from $\boldsymbol{q}$ according to a distribution $p(\boldsymbol{q}_i|\boldsymbol{q})$. We use the following deterministic sampling methods for generating variants of the original query. Each method corresponds to a different set of assumptions about the nature of the query, such as which aspects are most important. From least to most sophisticated, these are as follows.

- *No expansion* (NoExp). Use only the original query. The assumption is that the given terms are a complete description of the information need.

- *Single term-at-a-time* (TAT). A single term is chosen from the original query. This assumes that only one aspect of the query, namely, that represented by the term, is most important.

- *Leave-one-out* (LOO). A single term is left out of the original query for each variant. The assumption is that one of the query terms is a noise term. The LOO strategy may be seen as a form of *jackknife* estimator [Efron 1979].

- *Sigma-point sampling* (SPS). A very general sampling formulation that combines the features of LOO and TAT in a continuously variable form. The key idea of sigma-point sampling is to choose a small number of points that approximate a query neighborhood density around the initial query. The sigma points are chosen such that their mean and variance are equal to the mean and variance of the query neighborhood distribution, which we define as a Dirichlet prior with sharpness parameter β_U. When $\beta_U \gg 1$, the variants are only small adjustments to the original query. If $\beta_U \ll 1$, we get a mixture of variants, half of which are very close to a LOO sample and the other half to a TAT sample[4]. Details on sigma-point sampling are given in Section 4.2.5 of Chapter 4.

The final result of running query variants $\boldsymbol{q}_i$ with document set resampling is a set of feedback models $\{\hat{\theta}_F^{\boldsymbol{q}_i}\}$.

3.1.6 Justification for a sampling approach to feedback

In addition to the general flexibility and efficiency reasons for sampling that were discussed in Section 2.2, the use of sampling is particularly apt for pseudo-relevance feedback for two reasons.

First, we want to improve the quality of individual feedback models by smoothing out variation when the baseline feedback model is unstable. In this respect, our approach resembles *bagging* [Breiman 1996], an ensemble approach which generates multiple versions of a predictor by making bootstrap copies of the training set, and then averages the (numerical) predictors. In our application, top-retrieved documents can be seen as a kind of noisy

[4]For queries of two words, we revert to LOO sampling.

training set for relevance, and the feedback algorithm is a soft relevance predictor for terms. Unlike traditional bagging, however, the individual relevance predictors for terms may also exhibit complex inter-term correlation structure, and so in theory this scenario also has connections to structured prediction.[5]

Second, sampling is an effective way to approximate basic properties of the feedback posterior distribution, which can then be used for improved model combination. For example, a model may be weighted by its prediction confidence, estimated as a function of the variability of the posterior around the model. We now discuss this idea in more depth.

3.2 Model Combination

We now discuss ways to combine the set of models $\{\hat{\theta}_F^{\mathbf{q}_i}\}$ into a final feedback model $\hat{\theta}_F$. In this chapter we apply a simple heuristic approach to model combination. In Chapters 4 – 6 we develop an alternative approach to model combination based on convex optimization that can exploit dependencies between terms.

Let $w(\theta_j)$ be the weight given to model $\hat{\theta}_j$, where $\sum_k w(\theta_k) = 1$. We have two options for model combination. The simplest is to assume that all models are independent, and that $w(\theta_j)$ is based on properties of that model alone. The second type is more complex, but uses the more realistic assumption that models may be correlated. This is sensible because in our applications, models are typically generated from similar input.

We first consider the case where models are independent. In that scenario, we make the assumption that terms within each model are also independent. We then discuss the dependent model case we give a simple but effective weighting method that uses inter-model correlation.

3.2.1 Model Combination: Independent models

Suppose we have M feedback models to combine. One widely-used model combination approach in other domains is to perform Bayesian model averaging among the component models. This is a linear mixture of the component feedback models with each model weighted by its posterior probability. This gives

$$w(\theta_j) = \frac{\mathcal{L}(\theta_j|\boldsymbol{q}) \cdot \tau_j}{\sum_k \mathcal{L}(\theta_k|\boldsymbol{q}) \cdot \tau_k} \tag{3.4}$$

[5] In this work, we only model weak negative correlation between terms via the use of a Dirichlet distribution. More interesting correlation structures using, for example, the logistic normal, are possible.

where $\mathcal{L}(\theta)$ is a likelihood function over feedback models conditioned on the query, and τ_i is the prior probability of model θ_i. As more evidence appears about which model is likely to be the 'correct' one, the posterior weights sharpen toward the 'best' model. Model averaging makes some strong assumptions: it assumes that one of the component models is the 'true' generative model, that posteriors are accurate, and generally that the components reflect exhaustive and mutually exclusive generative models of the data.

If we assume that the model most likely to be the 'true' model is the query model $\theta_{\boldsymbol{q}}$ estimated from the original $\boldsymbol{q}$, then using the loss function $L(\theta) = KL(\theta_{\boldsymbol{q}}||\theta)$, we obtain a form of Bayesian extension [Akaike 1979] to the Akaike Information Criterion [Akaike 1974]. The AIC was originally introduced for model selection. Instead of selecting a single model, however, the Bayesian extension of AIC averages over model choices, using $exp(-\frac{1}{2}AIC)$ for the role of model likelihood. Using the loss function $L(\theta)$ above and neglecting the constant model dimension factor of AIC, this gives

$$\mathcal{L}(\theta_j|\boldsymbol{q}) \propto \exp -\frac{1}{2}KL(\theta_{\boldsymbol{q}}||\theta_j) \tag{3.5}$$

as an approximate expression for the model likelihood, assuming a uniform prior $\tau_i = 1/M$. This in turn gives the model weight

$$w(\theta_j) = \frac{\exp -\frac{1}{2}KL(\theta_{\boldsymbol{q}}||\theta_j)}{\sum_k \exp -\frac{1}{2}KL(\theta_{\boldsymbol{q}}||\theta_k)}$$

which gives models more weight that are relatively 'closer' to the most likely model. Uniform weighting with $w(\theta_j) = \frac{1}{M}$ is another possible choice.

For a richer hypothesis space we can move beyond interpolation of models, where the mixture weight for each component of the parameter vector θ_i is equal to the model weight. When the vector components represent term weight, this amounts to *term-specific* model weights. There are two cases to consider: we treat terms as independent or we consider correlations.

Independent terms

In the simplest case, we can consider each term w independently of the others. We call this our 'independent term' model combination strategy that ignores dependency relations between terms. Our goal is to find optimal parameters $p(\mathcal{R} \mid w)$ for a final feedback model

$\hat{\theta}_{Final}$ by using evidence from all sources available: the original query, and query variants with their corresponding feedback submodels $\hat{\theta}_j$. Defining the optimality criteria is a critical step and will determine the parameter estimation method.

We estimate two parameters for each term. First, we estimate a *relevance* probability $\mu_w = p(\mathcal{R} \mid w)$. Second, we estimate a *variance* factor σ_w^2 giving the variance that μ_w has between the given M feedback models, normalized by the within-model variance. We define the heuristic

$$\sigma_w^2 = \sum_{i=1}^{M} \lambda_i \cdot \frac{(\mu_{w,i} - \mu_w)^2}{\sigma_{w,i}^2} \tag{3.6}$$

where λ_i is the overall weight assigned to model i (from Section 3.2.1), and $\sigma_{w,i}^2$ is the variance of term w within model i according to the Dirichlet distribution $\boldsymbol{\alpha_i}$ estimated for model i during document set resampling.

Optimality criteria can be very different depending on the context of model combination. In the case of query variants, we expect term probabilities in the corresponding submodels to vary in response to query changes. Higher variance terms are better because they are more highly correlated with query terms. On the other hand, when combining models from resampled document sets, *lower* within-model term variance is better, because we assume a stable latent relevance model that generated the submodels. Both of these factors are present in this model combination heuristic.

Independent models, dependent terms

In this scenario, we would model dependencies (such as co-occurrence) between terms within each model. The computational cost is much higher: language model vocabularies tend to have at least 10,000 terms and the number of dependencies grows as the square of vocabulary size. This in turn increases our need for training data to fit the much larger number of parameters in the covariance matrix. There are ways to mitigate this expense, such as restricting the vocabulary size, or the complexity of covariance structure we can express.

One type of flexible structure restriction we have investigated in a different study is a graph-based term dependency language model [Collins-Thompson & Callan 2005] in which a term's probability is derived from approximating the stationary distribution of a lazy random walk on the graph. The graph is built from multiple sources of evidence about terms.

We do not apply term dependencies for model combination in this chapter. Instead, we explore that in Chapter 6, by creating a term covariance matrix to form a risk minimization objective. The covariance matrix is derived using a more efficient method than [Collins-Thompson & Callan 2005] based on the perturbation kernel of Chapter 4.

3.2.2 Model Combination: Dependent models

While treating sampled models as independent makes for simpler analysis, it is not especially realistic. In the case of pseudo-relevance feedback, the feedback models generated after running slightly different query variants are likely to be somewhat correlated. This implies that when the feedback models are used to predict relevance, their successes and errors will also be correlated. In some scenarios, ignoring this correlation may increase risk, because we are making redundant bets that will either be 'very right' or 'very wrong' together. On the other hand, it may pay to reward models that are part of a highly correlated set (a cluster) while downweighting outliers. In any case, a better approach is to consider the risk of the sampled models *as a set* instead of individually. This means estimating the model weights w_i according to the interaction of the θ_i. As a specific example, suppose we have several similar low-confidence feedback models, and a single very different high-confidence model. For a given term, this means we may have several low-confidence relevance weights from very similar sources, and a high-confidence single-source relevance weight. If we ignored the overall similarity between the component sources the result would be biased towards a majority-vote decision in favor of the low-confidence classifiers. By taking model dependence into account, the low-confidence classifiers would be discounted by a factor dependent on their correlation [Ghahramani & Kim 2003].

As one example of a scheme for calculating the weighting of dependent models, we apply the technique from Monte Carlo integration described in Chapter 2 called *multiple importance sampling* using the *balance heuristic* [Veach 1997]. This scheme weights each model according to its relative probability compared to the other models. This assumes that we can define a density $p_j(\cdot)$ from which model $\hat{\theta}_j$ was drawn (typically, as the most likely observation), and that we can calculate the probability $p_j(\hat{\theta}_k)$ of one model $\hat{\theta}_k$ in the density *of any other model* $\hat{\theta}_j$.

More specifically, suppose we assume the j-th model $\hat{\theta}_j$ to be the mode of a density $p_j(\cdot)$ having a Dirichlet distribution with parameters $\boldsymbol{\alpha_j}$. In particular, let s_j be the scale factor for $\boldsymbol{\alpha_j}$. Under these assumptions, the probability mass at the mode $c_j = p_j(\hat{\theta}_j)$ is

like a confidence score for model $\hat{\theta}_j$: when $p_j(\cdot)$ is sharply peaked, the probability mass is highly concentrated around the mode, and c_j will be close to 1. When the Dirichlet distribution is spread out and uncertain, the mode accounts for much less mass and so c_j will be much smaller. We also define $var(\hat{\theta}_j) = 1/c_j$ as the variability of $\hat{\theta}_j$, which is high when confidence is low. Now consider a different model $\hat{\theta}_i$. We can define a similarity measure

$$\sigma(\hat{\theta}_i, \hat{\theta}_j) = p_j(\hat{\theta}_i) \tag{3.7}$$

Using the balance heuristic [Veach 1997] for model combination, the weight $w(\theta_j)$ given to model $\hat{\theta}_j$ is given by

$$w(\theta_j) = \frac{n_j p_j(\hat{\theta}_j)}{\sum_k n_k p_k(\hat{\theta}_j)} \tag{3.8}$$

With all $n_j = 1$, we can rewrite the above expression in terms of confidence and similarity measures just discussed, namely

$$w(\theta_j) = \left(1 + var(\hat{\theta}_j) \cdot \sum_{k \neq j} \sigma(\hat{\theta}_j, \hat{\theta}_k)\right)^{-1}. \tag{3.9}$$

Thus, the balance heuristic weight $w(\theta_j)$ assigned to model $\hat{\theta}_j$ accounts for both individual model confidence and cross-model correlation. A feedback model that is similar to many other models will have a high similarity factor $\sum_{k \neq j} \sigma(\hat{\theta}_j, \hat{\theta}_k)$. An increasing similarity factor will decrease the model weight, reflecting the fact that we are making redundant 'bets' that may increase our risk. Moreover, this effect is amplified when $var(\hat{\theta}_j)$ is high (model confidence is low), which is exactly what we need. In the special case where all models are totally dissimilar, the similarity factor will be zero and all models are weighted equally.

Using the fact that the $p_j(\cdot)$ are Dirichlet, Eq. 3.9 can be written as

$$w(\theta_j) = \left(1 + \frac{1}{s_j} \cdot \sum_{k \neq j} e^{-JS(\hat{\theta}_j \| \hat{\theta}_k)}\right)^{-1} \tag{3.10}$$

This completes the discussion of our framework for pseudo-relevance feedback using sampling. We now introduce several measures for evaluating the robustness of information retrieval algorithms.

3.3 Evaluation Methods for Robustness

When applying a query expansion algorithm reduces precision compared to the initial query, we say that the algorithm hurts that query and call this a *failure* of the expansion algorithm. Maximizing the *worst-case* performance of an expansion algorithm means minimizing the number and/or magnitude of failures. A *robust* algorithm is one that has both good worst-case performance and good average performance. Ideally, a robust feedback method would never perform worse – and hopefully, better – than using the original query. To evaluate robustness of query expansion algorithms, we use three approaches that summarize different aspects of an algorithm's worst-case performance, and the tradeoff between worst-case performance and overall performance: the *robustness index*, *robustness histograms*, and *risk-reward tradeoff curves*.

3.3.1 Robustness Index

As a general summary statistic for robustness we employ a very simple measure called the *robustness index* (RI). [6] For a set of queries Q, the RI measure is defined as:

$$RI(Q) = \frac{n_+ - n_-}{|Q|} \tag{3.11}$$

where n_+ is the number of queries helped by the feedback method and n_- is the number of queries hurt. Here, by 'helped' we mean obtaining a higher (non-zero gain in) average precision after applying feedback. Different flavors of RI are possible, using P20, break-even point, and others instead of average precision. We focus on average precision since this is a widely-used measure of retrieval effectiveness for generic IR evaluations. However, we do also measure top-20 robustness results using a more sensitive measure, R-Loss@20, described below in Section 3.3.3.

The range of RI values runs from a minimum of -1.0, when all queries are hurt by the feedback method, to $+1.0$ when all queries are helped. A major drawback of the RI measure is that it ignores the actual *magnitude* of improvement or decline across the set Q of queries[7]. However, we use it because it is easy to understand as a general indication of robustness.

[6] This is sometimes also called the *reliability of improvement index* and was used in Sakai*et al.* [Sakai et al. 2005].

[7] A paired or t-test could be used to partly account for magnitude of changes.

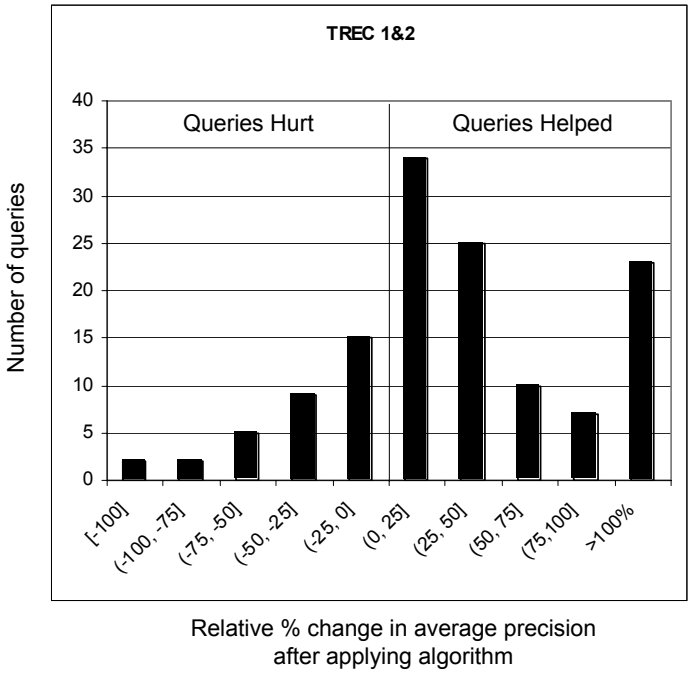


Figure 3.4: Example of a histogram showing the distribution of gains and losses in MAP over a set of queries, as a result of applying a particular query expansion algorithm.

3.3.2 Robustness histograms

For a more detailed look at an algorithm's effectiveness, we can plot the distribution of gains and losses for individual queries as a result of applying the algorithm. Typically we use a statistic such as percentage MAP gain or decrease after applying the expansion algorithm. The performance range (for the loss or gain in the statistic) is usually grouped into bins in increments of 10% or 25% (as in the example) change in the statistic. The *y*-axis gives the number of queries that fall into each bin. An example of a robustness histogram is shown in Figure 3.4. Unlike the single RI statistic, histograms can distinguish between two systems that might hurt the same number of queries but which do so by very different magnitudes.

A histogram gives results for a particular fixed algorithm, or choice of algorithm parameters, such as the feedback interpolation parameter α. A robustness histogram can capture some notion of the tradeoff between the *downside risk* of an algorithm, as measured by the area of the bins on the left half of the graph ('queries hurt'), versus the overall *reward* as measured by the average gain over all the bins on the histogram.

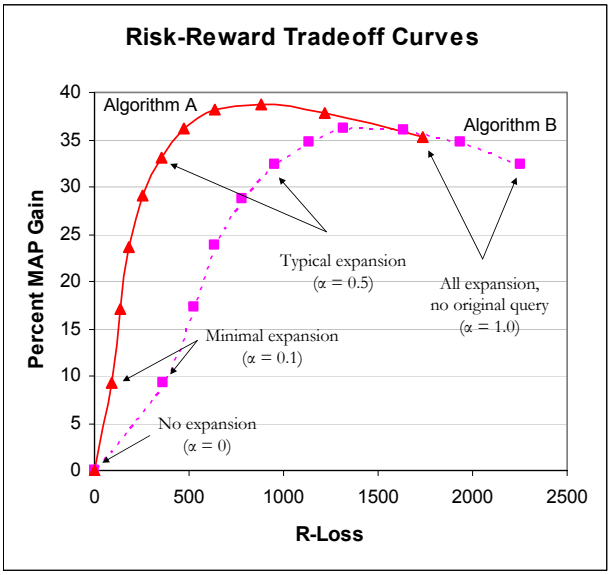


Figure 3.5: Typical risk-reward tradeoff curve for two algorithms, showing how downside risk (R-Loss) and MAP improvement change together as the feedback interpolation parameter α is increased from 0. (original query, no expansion) to 1.0 (all feedback model, no original query). Curves that are *higher* and *to the left* give a better tradeoff.

What the histogram does not show is how the risk-reward tradeoff *changes* with change in an algorithm parameter. Typically, we are concerned with the interpolation parameter α as it varies from 0 to 1. For that, we need the following very useful *risk-reward tradeoff curve*.

3.3.3 Risk-reward tradeoff curves

We observe that when interpolating a feedback model with the original query model there is generally a risk/reward tradeoff: giving more weight to the original query model (lower α) is less risky but also reduces the potential gains when the feedback model is effective, and vice versa. By plotting the joint risk and reward that the model achieves over a range of α values, we obtain a *curve* that gives a more complete picture of the quality of the feedback model as the interpolation parameter α given in Eq. 3.1 is varied from $\alpha = 0.0$ (original query only) to $\alpha = 1.0$ (all feedback).

Machine learning, text classification, and information retrieval evaluations have previously used certain curves plotting the relation between two variables related to system effectiveness. In the 1960s, Swets [Swets 1963] introduced the use of signal detection the-

ory to IR evaluation in the form of ROC curves, plotting false positives *vs.* false negatives. Another early method, still widely-used, is the precision-recall curve (or P-R curve) [Manning & Schütze 2000]. Recently, deeper connections between ROC and P-R curves have been shown [Davis & Goadrich 2006].

Risk/reward tradeoff curves were first introduced by Markowitz [1952] as part of his pioneering finance work on portfolio selection (for which he received a Nobel Prize). Markowitz identified many of the key features and analysis methods for risk-reward curves that are still used in economics and finance today. In this section we show how these concepts can also add a new dimension to the analysis of information retrieval algorithms.

Measures of risk and reward

To compute a risk-reward tradeoff curve for an information retrieval algorithm, we must first decide on how to quantify risk and reward. The appropriate measures will vary depending on the type of retrieval task: a good "reward" measure for Web search, for example, may be precision in the top-20 documents (P20); legal IR applications may focus on recall; and general IR evaluations may use mean average precision (MAP). We generally will focus on risk-reward curves using MAP or P20 as the "reward" measure, and this is plotted on the *y*-axis of the chart.

The "risk" measure is meant to capture the variance or some other undesirable aspect of the "reward" measure that should be minimized. To evaluate query expansion algorithms, we assume the results from the initial query represent our minimal acceptable retrieval performance: we do not want to obtain worse results than the initial query. We are therefore particularly interested in the *downside risk* of an algorithm: the reduction in reward due to failure cases. (Recall that a failure case is one in which the reward after using the algorithm is lower than the reward obtained with the initial query.) The risk measure is assigned to the *x*-axis of the risk-reward curve.

We denote $R_I(Q)$ as the initial reward obtained with the query Q with no expansion, and $R_F(Q)$ as the final reward obtained when a query expansion algorithm is applied to Q. We denote the test set of queries as $\boldsymbol{Q}$ and the set of queries for which the expansion algorithm fails as $\boldsymbol{Q}_{FAIL}$. Then the downside risk $F_{FAIL}(Q)$ for Q is simply

$$F_{FAIL}(Q) = \begin{cases} R_I(Q) - R_F(Q) & \text{if} \quad R_I(Q) - R_F(Q) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

and the total downside risk for the test set of queries is

$$F_{FAIL}(\mathcal{Q}) = \sum_{Q \in \mathcal{Q}} F_{FAIL}(Q) \tag{3.13}$$

When the reward measure is precision at the top k documents – such as P20 – we define a derived quantity called *R-Loss at k*, denoted RL_k, that is the *net loss of relevant documents due to failure*. Since precision is the fraction of the k documents that are relevant, RL_k is simply

$$RL_k(Q) = k \cdot F_{FAIL}(Q) \tag{3.14}$$

The same definition can be applied when the reward measure is MAP and we refer to this simply as *R-Loss* (without specifying k). In this case, we set k to the size of the retrieved document set, which is $k = 1000$ unless otherwise specified. Just as MAP gives a combined picture of precision results averaged over multiple values of k, so the R-Loss measure gives an averaged net loss of relevant documents due to failure.

Examples of typical risk-reward curves for query expansion algorithms are shown in Figure 3.5. The curve is generated by varying the interpolation parameter α. Because the reward is relative to the initial query, all curves start at the origin ($\alpha = 0$). We will typically plot in α increments of 0.1.

Properties of risk-reward curves

A higher-quality feedback model will give tradeoff curves that are consistently higher and to the left of the baseline model's tradeoff curve. We say that one tradeoff curve *A dominates* another curve *B* if the reward achieved by *A* for any given risk level is always at least as high as achieved by *B* at the same risk level. Thus, we say that one query expansion algorithm *A dominates* another algorithm *B* if the tradeoff curve for *A* dominates the tradeoff curve for *B*. For example, in Figure 3.5 algorithm *A* dominates algorithm *B*.

Each point on the risk-reward chart represents the outcome of an experiment averaged over (typically) dozens or hundreds of queries, for a particular setting of algorithm parameters. If we could run the algorithm in an unlimited number of experiments using every possible combination of parameter settings, we would obtain a (large!) set of points scattered over the chart. The *efficient frontier* on a risk-reward graph is the boundary of the convex hull containing these points and represents the best performance that an algorithm can achieve at any given level of risk, for any choice of parameters. Typically, the efficient

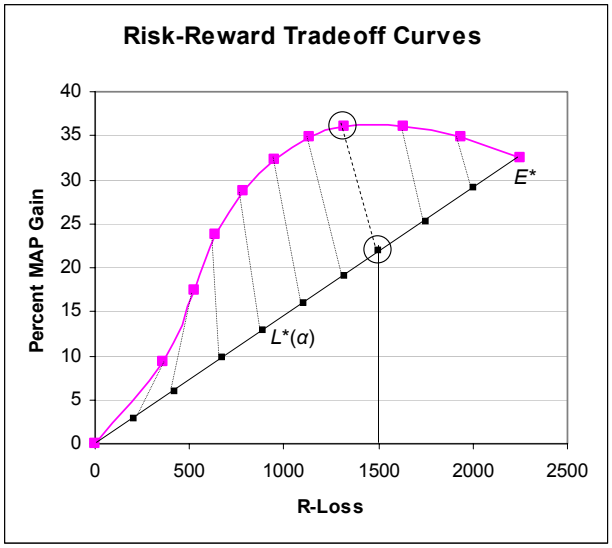


Figure 3.6: Example showing the information retrieval equivalent of the two-fund theorem from finance: how an effective α can be found for a given level of risk

frontier must be approximated by sampling a range of parameter choices.

The *risk-reward ratio* $\rho(P)$ of a feedback strategy (point) P that achieves MAP gain $G(P)$ and R-Loss $F(P)$ is simply

$$\rho(P) = G(P)/F(P) \tag{3.15}$$

which is the *slope* of the line joining P to the origin.

A number of potentially useful measures follow from this view. For example, we could calculate the risk-reward function $\rho(\alpha)$ as a function of α, and choose $\alpha = 0.5$ as the standard ratio for an algorithm, the *midpoint risk tradeoff*, giving a single value that could be used to compare with other algorithms on the same collection.

Borrowing another concept from portfolio theory, the *Sharpe ratio* is the slope of the point $P^\star$ on the algorithm's efficient frontier with maximum slope $\rho^\star$. [8] Thus, using some approximation of the efficient frontier (maintaining a convex hull), we could identify the *best achieved tradeoff* of an algorithm.

A two-step heuristic for effective linear interpolation.

Suppose we know the level of risk F we want to accept for a set of queries (say, in terms of average relevant documents lost to failures in the top k). We have an expansion algorithm with a tunable parameter set Θ that produces a feedback model θ, and we can combine the initial query $\boldsymbol{q}$ with the usual linear interpolation parameter α controlling the mixture between $\boldsymbol{q}$ and the feedback model θ. How should we optimize the joint parameter set (Θ, α) to to achieve the given level of risk F on average?

To answer this question, we first note that the line $L(\alpha)$ joining the two end points (corresponding to $\alpha = 0$ and $\alpha = 1.0$) of the tradeoff curves in our experiments often provides a lower bound for the entire tradeoff curve. Furthermore, tradeoff curves are concave or very close to concave[9], so that the line joining any two points on the tradeoff curve gives the minimum expected performance for the curve above the line. This observation suggests that the search for an optimal interpolated model for any expansion algorithm at a particular level of risk can be broken into two distinct steps, as illustrated in Figure 3.6.

1. Optimize the risk-reward tradeoff of the expansion model by itself (i.e. at $\alpha = 1.0$) to obtain the optimal point $E^\star$. This is equivalent to maximizing the slope of $L^\star(\alpha)$, the line joining the origin to $E^\star$.

2. Choose α to achieve the specified risk level on $L^\star$. The actual MAP gain will be at least as high as the MAP gain on $L^\star$ at α. Use this value of α to interpolate with the initial query model.

Because the actual tradeoff curve is not linear, we are making the assumption that optimizing $L^\star$ will result in optimizing the MAP gain for a given risk level on the rest of the curve. Comparing the baseline and QMOD tradeoff curves, we see that shifting the endpoint $E^\star$ up and left results in a corresponding shift of the entire curve up and left (by varying amounts at different α points). Our two-step observation is a conjecture, but the observed behavior of all tradeoff curves we have seen strongly suggests this is a useful heuristic. This method can simplify the search for an optimal model at a given level of risk, and can be used with any feedback algorithm.

This heuristic is similar to a basic result of modern portfolio theory known as the *two-fund separation theorem* [Tobin 1956]. If investors care only about the expected return

[8] In finance, a point P represents a choice of portfolio, and $P^\star$ is called the "market portfolio".

[9] See Figure 3.8 and other MAP/R-Loss curves for some evidence for this.

and the standard deviation of their portfolio return, then every investor holds a portfolio consisting of the market portfolio M and the risk-free asset in some proportion. In information retrieval terms, the risk-free asset is the original query, and the market portfolio M corresponds to the feedback model $P^\star$ on the efficient frontier with the highest Sharpe ratio.

There are important differences in the properties of risk-reward curves between information retrieval and finance. In finance, the optimal tradeoff solutions for a mixture of a risk-free asset and a given portfolio P, parametrized by α, the proportion invested in P, form a *straight line* joining the corresponding points on the risk-reward chart.[10] This linearity results from the quadratic nature of the traditional mean-variance utility function. However, as the query expansion curves in Section 3.4 make clear, the best risk and return tradeoffs available for an algorithm do *not* necessarily co-vary linearly as the interpolation weight α shifts toward the "risk-free" asset (query) and away from the feedback model $\hat{\theta}_F$. Instead, the curve is usually concave and above the line joining the origin (risk-free query) to the chart point corresponding to the feedback model. The cause of this requires further study, but we conjecture that this non-linearity is a consequence of the clustering behavior of relevant documents (and perhaps the nature of the mean average precision reward function).

3.4 Evaluation Results

In this section we present results confirming the usefulness of using sampling for pseudo-relevance feedback. In particular, we show that finding the mode of the feedback model distribution from weighted resampling of top-ranked documents, and of combining the feedback models obtained from different small changes in the original query, results in retrieval that is both more precise and robust than the baseline method alone.

The results are organized as follows. In Section 3.4.3 we look at P20 (top-20 precision). Section 3.4.7 examines the effect of the number of samples on precision improvements. We use the evaluation methods described in Section 3.3 to assess robustness in Section 3.4.4. Section 3.4.8 compares different methods of generating query variants, while Section 3.4.9 compares two document sampling strategies. Finally, Section 3.4.10 analyzes how combining multiple models improves precision by smoothing out noise terms, including stopwords.

[10] In finance this is known as the *capital allocation line*.

3.4.1 General method

We evaluated performance on six TREC topic sets, covering a total of 700 unique queries. These TREC topic sets are TREC 1&2, TREC 7, TREC 8, wt10g, robust 2004, and gov2. Details on TREC topic sets, collections and methodology are given in Appendix C. We chose these corpora for their varied content and document properties. For example, wt10g documents are Web pages with a wide variety of subjects and styles while TREC-1&2 documents are more homogeneous news articles. Indexing and retrieval was performed using the Indri system in the Lemur toolkit [Metzler & Croft 2004] [Lemur 2002]. Our queries were derived from the words in the title field of the TREC topics. Phrases were not used. To generate the baseline queries passed to Indri, we wrapped the query terms with Indri's `#combine` operator. For example, the initial query for topic 404 is:

```
#combine(ireland peace talks)
```

We performed Krovetz stemming for all experiments. Because we found that the baseline (Indri) expansion method performed better using a stopword list with the feedback model, all experiments used a stoplist of 419 common English words. However, an interesting side-effect of our resampling approach is that it tends to remove many stopwords from the feedback model, making a stoplist less critical. This is discussed further in Section 3.4.10.

3.4.2 Baseline feedback method

For our baseline expansion method, we use an algorithm included in Indri 2.2 as the default expansion method. This method first selects terms using a log-odds calculation described by Ponte [Ponte 2000], but assigns final term weights using Lavrenko's relevance model[Lavrenko 2004].

We chose the Indri method because it gives a consistently strong baseline, is based on a language modeling approach, and is simple to experiment with. In a TREC evaluation using the GOV2 corpus [Collins-Thompson et al. 2004], the method was one of the top-performing runs, achieving a 19.8% gain in MAP compared to using unexpanded queries. In this evaluation, it achieves an average gain in MAP of 14.4% over the six collections.

Indri's expansion method first calculates a log-odds ratio $o(v)$ for each potential expansion term v given by

$$o(v) = \sum_D \log \frac{p(v|D)}{p(v|C)} \tag{3.16}$$

```
#weight(0.5 #combine(ireland peace talks) 0.5
#weight(0.10 ireland 0.08 peace 0.08 northern ...)
```

Figure 3.7: An example of a simple expanded query for TREC topic 404, showing the original query terms and expansion term set each given weight of $\alpha = 0.5$.

over all top-k documents D containing v, within collection C. Then, the expansion term candidates are sorted by descending $o(v)$, and the top m are chosen. Finally, the term weights $r(v)$ used in the expanded query are calculated based on the Relevance model

$$r(v) \propto \sum_{D} p(q|D)p(v|D) \tag{3.17}$$

The quantity $p(q|D)$ is the probability score assigned to the document in the initial retrieval. We use Dirichlet smoothing of $p(v|D)$ with $\mu = 1000$.

This relevance model is then combined with the original query using linear interpolation, weighted by a parameter α. By default we used the top 50 documents for feedback and the top 20 expansion terms, with the feedback interpolation parameter $\alpha = 0.5$ unless otherwise stated. For example, the baseline expanded query for topic 404 is shown in Figure 3.7

3.4.3 Expansion precision performance

For each query, we obtained a set of B feedback models using the Indri baseline. Each feedback model was obtained from a random sample of the top k documents taken with replacement. For these experiments, $B = 30$ and $k = 50$. Each feedback model contained 20 terms. On the query side, we used leave-one-out (LOO) sampling to create the query variants, since as we show later, LOO sampling generally dominated the other methods for all collections. We estimated an enhanced feedback model from the Dirichlet posterior distribution for each query variant, and used the variance model combination heuristic to obtain term weights for the combined feedback model from all the query variants. We call our method *resampling feedback* using heuristic model combination and denote it as RS-FB here (later, we also refer to this as HMC RS-FB if comparing model combination methods) We denote the Indri baseline feedback method as Base-FB. Results from applying both the baseline expansion method (Base-FB) and resampling expansion (RS-FB) are shown in Table 3.1. These results use per-term model weights, with uniform model-wide priors.

Collection		NoExp	Base-FB ($\alpha = 0.5$)	RS-FB ($\alpha = 0.5$)
TREC 1&2	MAP	0.1762	0.2317 (+31.9%) $^{\mathbf{N}}$	**0.2472** (+40.3%) $^{\mathbf{N,E}}$
	P20	0.4217	0.4483 (+6.9%) $^{\mathbf{N}}$	**0.4990** (+18.3%) $^{\mathbf{N,E}}$
	R-Loss@20	0/366	117/366 (-32.0%)	**64/366** (-17.5%)
	RI	0	0.4844	**0.5781**
TREC 7	MAP	0.1830	0.2079 (+13.8%) $^{\mathbf{N}}$	**0.2165** (+18.3%) $^{\mathbf{N,E}}$
	P20	0.3456	0.3467 (+0.3%)	**0.3656** (+5.9%) $^{\mathbf{N,E}}$
	R-Loss@20	0/57	**23/57** (-40.4%)	24/57 (-42.1%)
	RI	0	0.4146	**0.4634**
TREC 8	MAP	0.1920	0.2220 (+15.5%) $^{\mathbf{N}}$	**0.2288** (+19.2%) $^{\mathbf{N,E}}$
	P20	0.3213	0.3585 (+11.8%) $^{\mathbf{N}}$	**0.3596** (+11.9%) $^{\mathbf{N}}$
	R-Loss@20	0/76	29/76 (-38.2%)	**23/76** (-30.2%)
	RI	0	0.4286	**0.4762**
wt10g	MAP	0.1747	0.1830 (+5.2%)	**0.1984** (+13.6%) $^{\mathbf{N,E}}$
	P20	0.2228	0.2340 (+5.4%)	**0.2494** (+11.9%) $^{\mathbf{N,E}}$
	R-Loss@20	0/158	59/158 (-37.3%)	**55/158** (-34.8%)
	RI	0	-0.0270	**0.1892**
robust2004	MAP	0.2152	0.2441 (+13.5%) $^{\mathbf{N}}$	**0.2538** (+17.9%) $^{\mathbf{N,E}}$
	P20	0.3252	0.3397 (+4.5%) $^{\mathbf{N}}$	**0.3538** (+8.8%) $^{\mathbf{N,E}}$
	R-Loss@20	0/394	124/394 (-31.2%)	**112/394** (-28.4%)
	RI	0	0.3364	**0.3818**
gov2 (2004–2006)	MAP	0.2736	0.2907 (+6.5%) $^{\mathbf{N}}$	**0.2959** (+8.1%) $^{\mathbf{N,E}}$
	P20	0.5214	0.5214 (+0.0%)	**0.5352** (+2.6%) $^{\mathbf{N,E}}$
	R-Loss@20	0/575	171/575 (-29.7%)	**126/575** (-21.9%)
	RI	0	0.0922	**0.1915**

Table 3.1: Comparison of baseline (Base-FB) feedback and re-sampling feedback using heuristic model combination (RS-FB). Precision improvement shown for Base-FB and RS-FB is relative to using no expansion. R-Loss changes are relative to no expansion (Base-FB), where negative change is good. For Robustness Index (RI), higher is better. Significant differences at the 0.05 level using the Wilcoxon signed-rank test are marked by **N** and **E** superscripts, for improvement over NoExp and Base-FB respectively.

The RS-FB method achieved consistent gains in precision over Base-FB for all six topic sets, measured by both MAP and P20. RS-FB obtained higher MAP and P20 than Base-FB for every topic set, giving a macro-averaged improvement of 9.9% over no expansion compared to 4.8% for Base-FB. The lowest P20 gain for RS-FB over NoExp was +2.6% for gov2 and the highest was +18.3% for trec12. For MAP, RS-FB achieved a macro-averaged gain in MAP of +19.6% over NoExp, compared to the Indri baseline expansion gain of +14.4%. Gains in both MAP and P20 over both no expansion and baseline expansion were statistically significant at the 0.05 level for virtually all precision measurements, according to a Wilcoxon signed-rank test. The lone exception was a single P20 measurement on TREC 8, which was equivalent to the baseline.

These gains in precision were accompanied by a universal *reduction* in expansion failures: RS-BF increased the Robustness Index over Base-FB for every topic set. Similarly, R-Loss@20, the actual net loss of relevant documents in the top 20, *decreased* compared to the baseline Indri expansion by amounts ranging from 6.7% (wt10g) to 45.3% (TREC 1&2). TREC 7 was the only topic set to show a small R-Loss increase (1 relevant document).

3.4.4 Evaluating Robustness

We now present figures using risk-reward tradeoff curves, and robustness histograms.

Risk/reward tradeoff curves

One obvious way to improve the worst-case performance of feedback is simply to use a smaller fixed α interpolation parameter, such as $\alpha = 0.2$, placing less weight on the (possibly risky) feedback model and more on the original query. We call this the 'small-α' strategy.

We compared the robustness trade-off curves between our resampling feedback algorithm, and the simple small-α method. We call the resampling feedback method HMC (for Heuristic Model Combination).

Tradeoff curves using MAP/R-Loss are summarized in Figure 3.8, and curves using P20/R-Loss@20 are in Figure 3.9. As expected, risk (as measured by R-Loss) increases continuously as we move along the curve, and MAP gain generally increases at first. At some breakeven point, MAP gain begins to drop as the original query model is given much less weight. Since higher and to the left is better, it is clear that for all six collections,

HMC resampling feedback gives a consistently dominant trade-off curve compared to the baseline feedback model, whether MAP or P20 is used as the reward measure.

Robustness Histograms

We examine the histogram of MAP improvement across sets of topics. (We examine both MAP and P20 improvement via trade-off curves in Section 3.4.4.) Relative changes in AP for all topics is given by the histogram in Figure 3.10. The number of queries helped or hurt by expansion is shown, binned by the loss or gain in average precision by using feedback.

Compared to Base-FB, the RS-FB method achieves a noticable reduction in the number of queries negatively affected by expansion (i.e. where AP is hurt by 25% or more), while preserving positive gains in AP. The results for TREC 1&2 are particularly good. RS-FB not only achieves higher MAP gain (Base-FB: +31.9%, RS-FB: +40.3%) but the robustness of RS-FB was superior : only 4 topics were hurt by 50% or more using resampling feedback, compared to 9 for the baseline method.

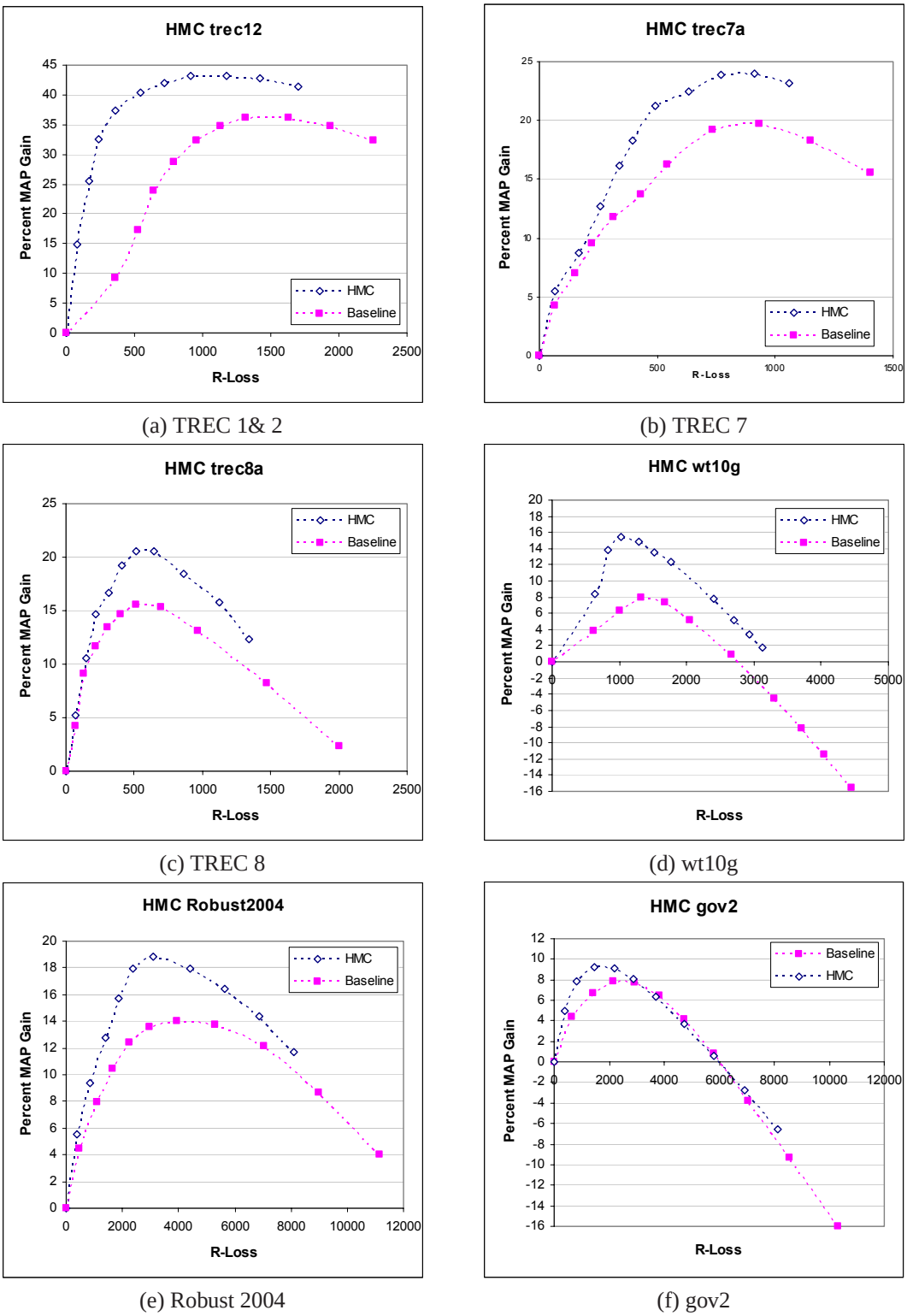
However, while these results are promising, there is room for improvement. There are still multiple failures at the -50% level and worse for all collections. In Chapter 6 we introduce an alternate model combination method that provides further reductions in serious failures. Section 6.4.2 has robustness histograms from that method for comparison.

3.4.5 Effect with an alternate expansion algorithm

To test the generality of RS-FB with another strong expansion baseline, we replaced the baseline Indri method (Relevance model) with a Rocchio-style vector space method in which the top k document vectors were given equal weight and used a *tf.idf* representation. The same query variants and document resampling were used as in the Indri experiments. The resulting tradeoff curves are shown in Figure 3.11. As with the Relevance model expansion baseline, RS-FB has strong performance and dominates the Rocchio *tf.idf* expansion for every collection.

3.4.6 Tolerance to poor baseline expansion algorithm

To test how tolerant RS-FB is to the choice of a very poor baseline algorithm, we replaced the default Indri method with a Rocchio scheme that ignores term frequency (tf) and uses only idf in the term representation. This results in a very noisy expansion model dominated by rare terms that are poor discriminators for relevance. The results for two representative collections, TREC 7 and wt10g, are shown in Figure 3.12. In both cases, this new *idf*



(a) TREC 1& 2

(b) TREC 7

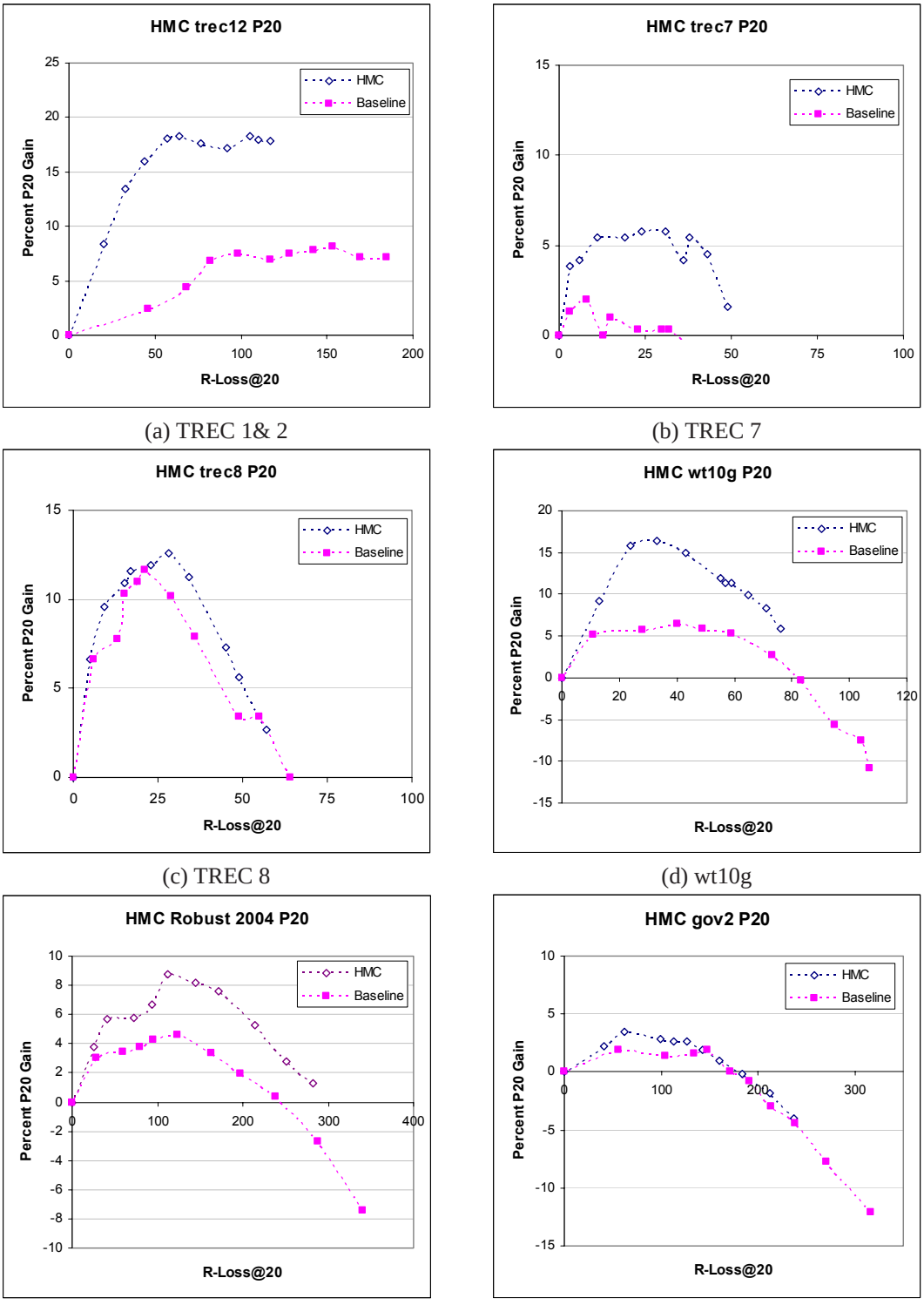
(c) TREC 8

(d) wt10g

(e) Robust 2004

(f) gov2

Figure 3.8: Risk-reward tradeoff curves for six TREC topic sets, showing how the HMC RS-FB robust feedback method consistently dominates the performance of the baseline feedback method. The baseline feedback model is the Indri Relevance Model. Tradeoff curves that are *higher and to the left* are better. Points are plotted in α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.

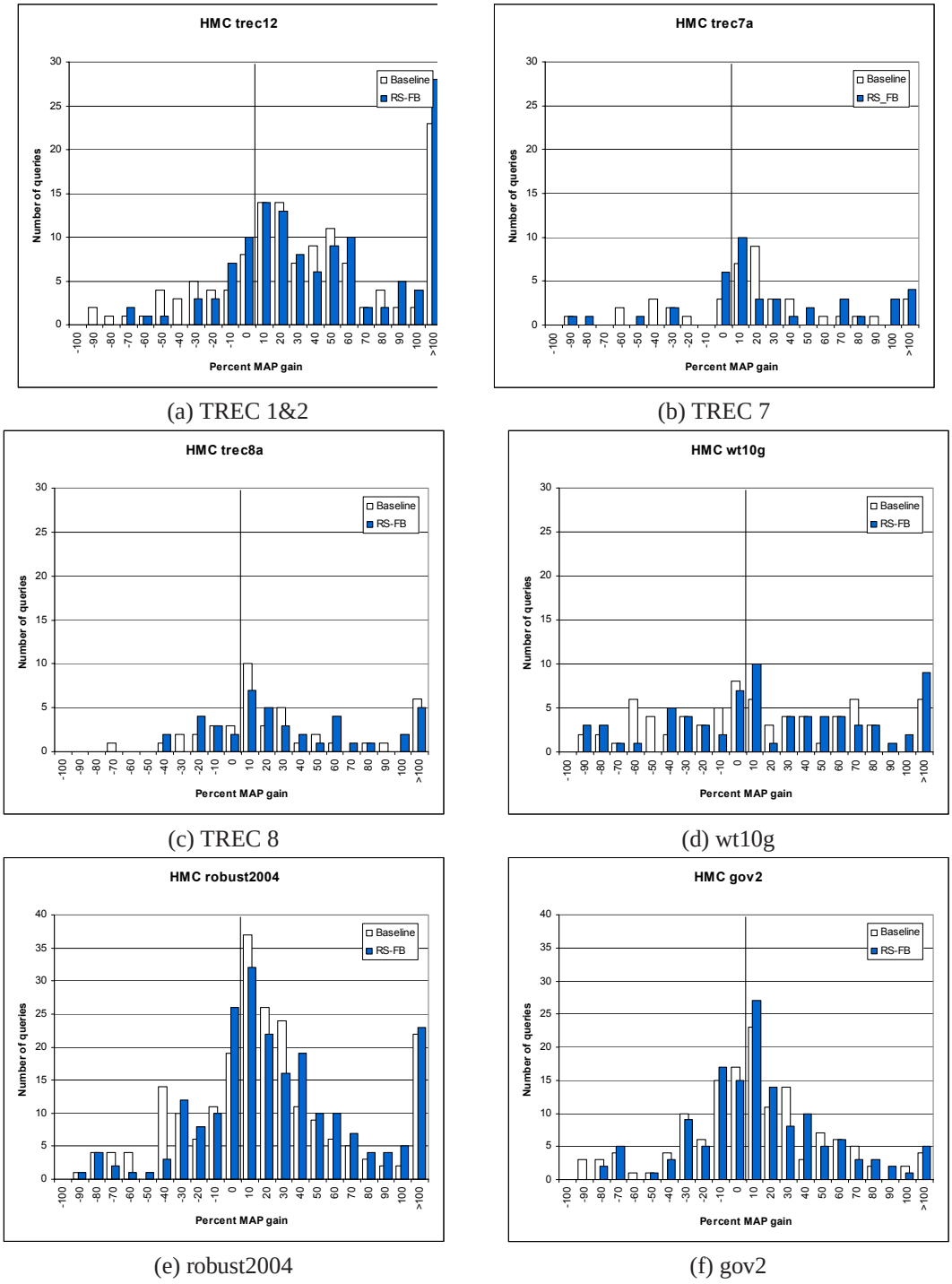


(a) TREC 1& 2 (b) TREC 7

(c) TREC 8 (d) wt10g

(e) Robust 2004 (f) gov2

Figure 3.9: Risk-reward tradeoff curves for six TREC topic sets using P20 and R-Loss@20 (instead of MAP and R-Loss). The baseline feedback model is the Indri Relevance Model. Tradeoff curves that are *higher and to the left* give a better risk-reward tradeoff. Curves are plotted with points at α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.

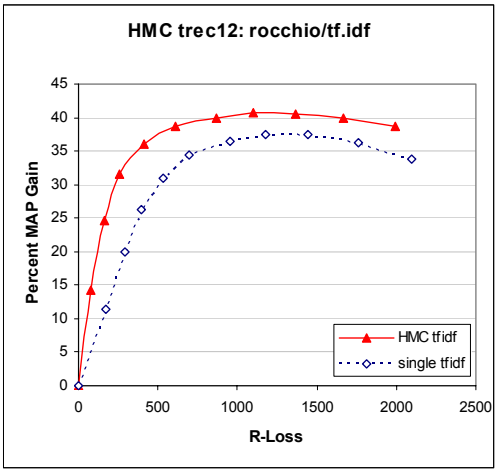


(a) TREC 1&2 (b) TREC 7

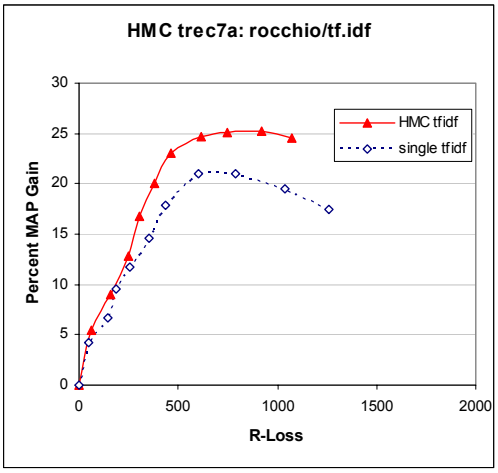
(c) TREC 8 (d) wt10g

(e) robust2004 (f) gov2

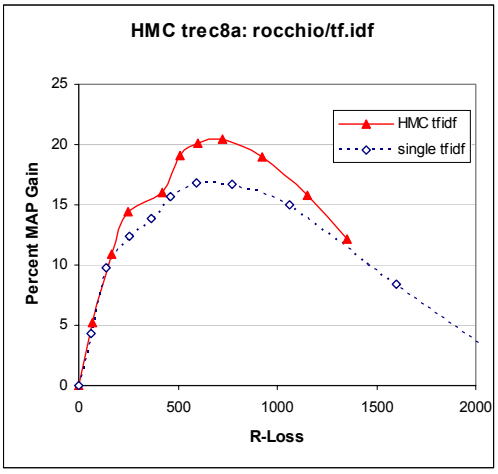
Figure 3.10: Robustness histograms for all six TREC collections, comparing the baseline expansion method (white) with the RS-FB resampling algorithm (solid). The number of queries helped or hurt by expansion is shown, binned by the loss or gain in average precision by using feedback. The baseline feedback here was Indri 2.2 (Modified Relevance Model with stoplist) and resampling feedback using both query (LOO) and top-document sampling.



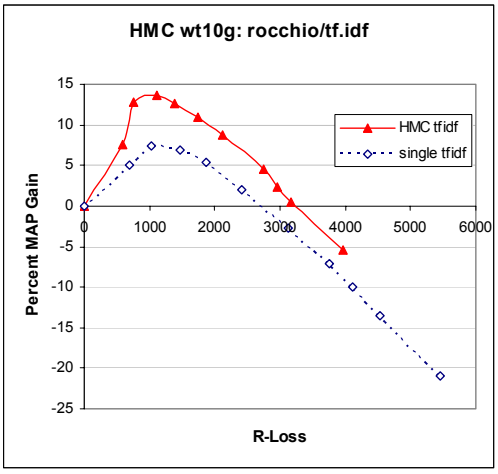
(a) TREC 1& 2



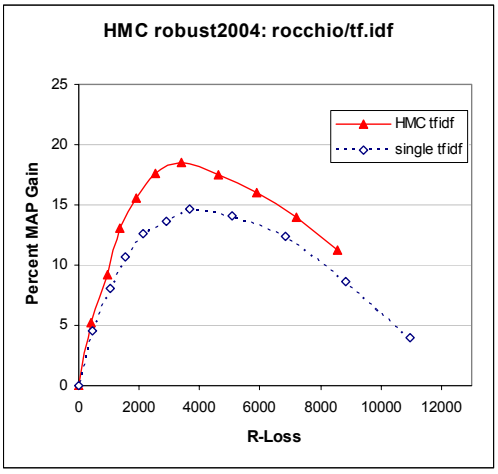
(b) TREC 7



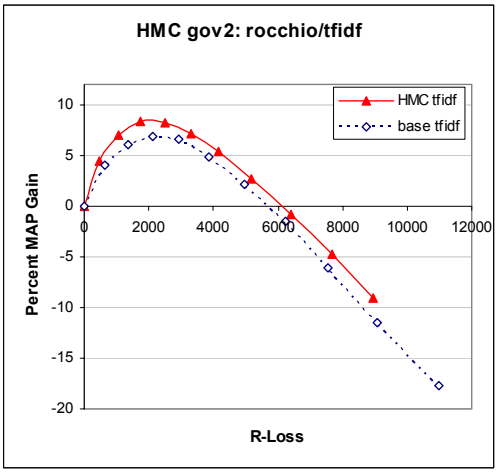
(c) TREC 8



(d) wt10g

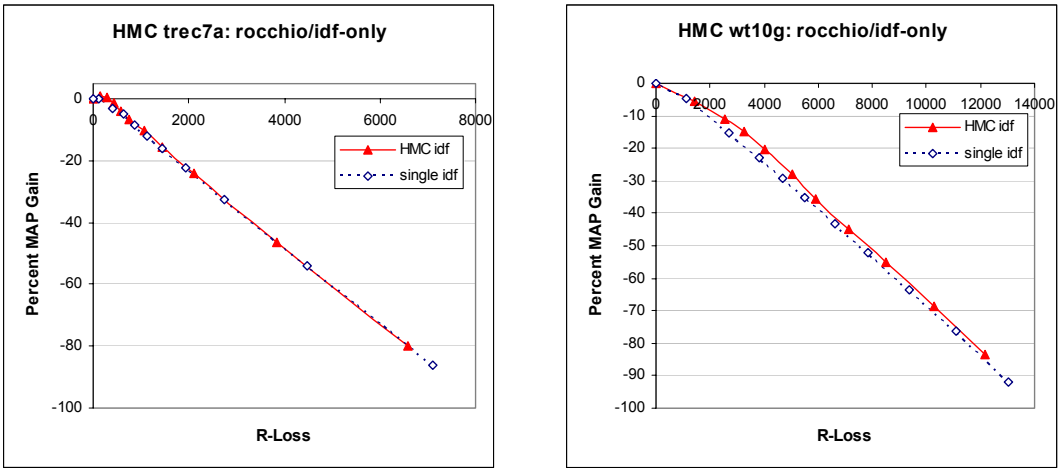


(e) robust2004



(f) gov2

Figure 3.11: The effect on risk-reward tradeoff curves of applying RS-FB (solid line) to an alternate, Rocchio-style expansion algorithm using *tf.idf* representation (dotted line) instead of the default Relevance model baseline. Tradeoff curves that are *higher and to the left* are better. Points are plotted in α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.



(a) TREC 7 (b) wt10g

Figure 3.12: Risk-reward tradeoff curves for two representative TREC topic sets, showing the effect of using RS-FB with a very poor baseline expansion algorithm. The solid line is the curve given by the RS-FB algorithm using the poor *idf* baseline. The dashed line is the curve given by the *idf* baseline alone. Results for other collections are similar.

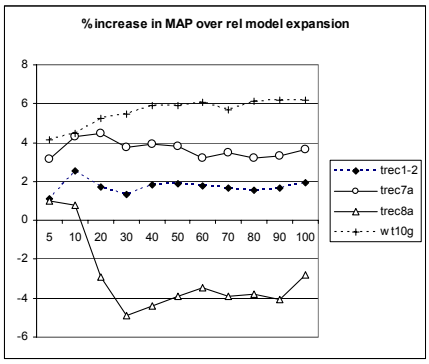
expansion algorithm performed terribly: MAP loss at $\alpha = 1.0$ was worse than -80% in all cases. Applying RS-FB to this poor baseline resulted in little improved performance, showing that the RS-FB method is not especially tolerant of a weak expansion baseline. We have omitted the other four standard collections because their results are similar.

However, in Chapter 6, we introduce a new, more selective model combination method that has, as one side-effect, much better tolerance of weak expansion algorithms. See Section 6.4.5 for a comparison.

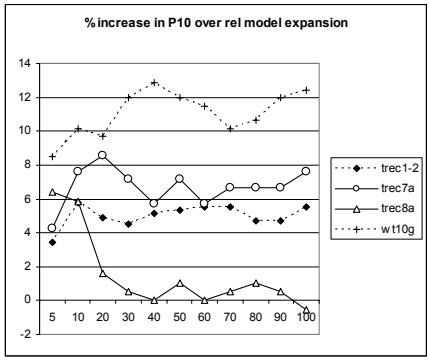
3.4.7 Effect of sample size

In Section 3.1.3 we described how we generate B feedback models, each from a different resampling of the top-retrieved documents $\mathcal{D}$. Because there is a moderate computational cost to computing a feedback model, it is important to understand the tradeoff between gains from the method and the number of feedback models computed. Here we evaluate the effect of B on MAP and P10 compared to the baseline feedback algorithm.

The results are summarized in Figure 3.13 for all collections. The left chart shows the stability of the MAP gain as B increases from 5 to 100, for a set of top-50 documents. MAP stabilizes after about 40 sampled document sets, although when $B \geq 20$ the MAP



(a) % MAP Gain over Baseline

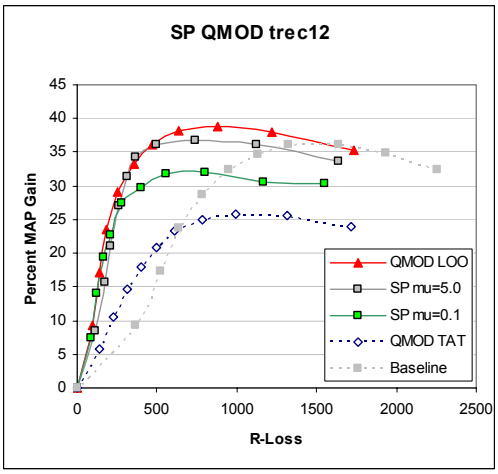


(b) % P10 Gain over Baseline

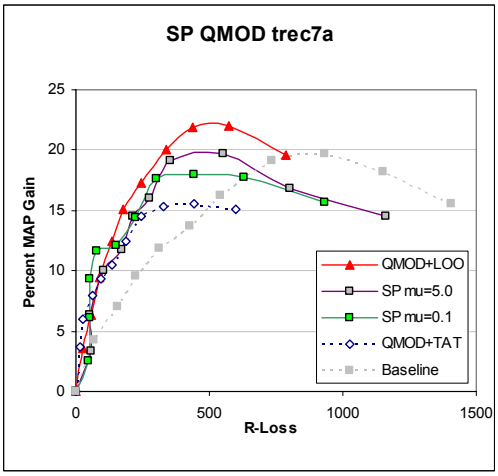
Figure 3.13: The stability of results as a function of the number of bootstrap samples from the top-retrieved documents. Gains or losses shown are relative to the Indri baseline expansion method.

differences are relatively minor. The story for P10 is similar. The majority of gains are achieved by the point $B = 10$, with the gain for wt10g actually declining to zero with more samples. The variance in the P10 chart is larger simply because the smallest possible increment of that statistic is ± 0.1. Interestingly, at $B = 5$ or $B = 10$, all collections showed a modest gain in both MAP and P10 over the baseline, while the wt10g collection showed significant worsening of both MAP and P10 for $B \geq 10$. The reason for this is likely as follows.

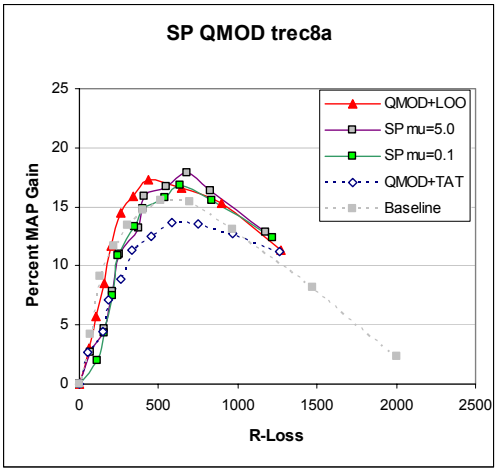
In a sense, B can be considered a smoothing parameter. As we take more resampled sets, we model the top-retrieved documents more and more accurately. Conversely, as B decreases, more noise is introduced into our model of the true feedback distribution, inducing a sort of generalization into the feedback model. It appears that even 5 resampled feedback models are enough to achieve small, consistent improvement in performance for all collections. Our choice of B thus depends on how we wish to trade off small but more consistent gains, with less computation, when $B \leq 10$, with the possibility for slightly larger potential gains for some collections but also higher variance and increased computational cost when $B \geq 10$. In general, a reasonable rule of thumb appears to be to take B to be about 20% of k, the number of top-retrieved documents.



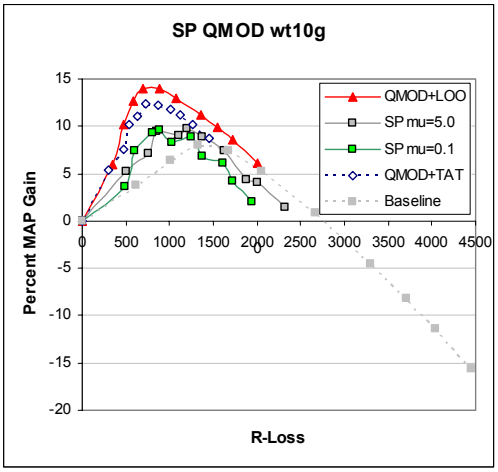
(a) TREC 1&2



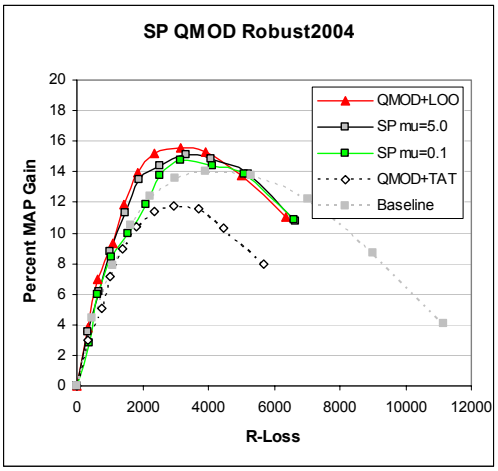
(b) TREC 7



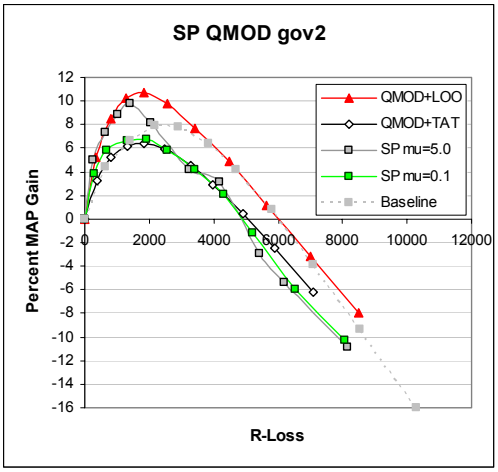
(c) TREC 8



(d) wt10g



(e) Robust 2004



(f) gov2

Figure 3.14: The effect of query variant sampling method on risk-reward tradeoff, showing how LOO sampling generally dominates the other methods. LOO is leave-one-out, TAT is term-at-a-time, and SP is sigma-point sampling. The baseline is Indri Relevance model expansion.

3.4.8 Effect of query sampling method

We compared the effect of TAT, LOO, and sigma-point query variant methods on the tradeoff curve of the baseline expansion algorithm. For sigma-point sampling (SP) we show two different runs, using $\beta_U = 0.1$ (wide) and $\beta_U = 5.0$ (more peaked) query neighborhood. The results are shown in Figure 3.14.

Leave-one-out (LOO) variants achieved the highest maximum MAP gain for every collection shown, and generally gave a dominant tradeoff curve over all other methods. Term-at-a-time (TAT) variants achieved the lowest MAP gain on all collections except wt10g. The LOO curve dominated the TAT curve for all six collections. This difference in performance between LOO and TAT is likely due to the fact that LOO preserves much more of the query context. There was little significant difference between SP $\beta_U = 5.0$ (query neighborhood more peaked around initial query) and SP $\beta U = 0.1$ (broader neighborhood). For the most 'well-behaved' collection, TREC 1&2, the peaked SP had consistently higher MAP gains for $\alpha > 0.5$ at equivalent risk levels. For the other collections, however, the methods performed comparably. Compared to the baseline, at $\alpha = 0.5$ using any query variant method led to a better risk-reward ratio for TREC 1&2, TREC 7, and wt10g, equal results for TREC 8 and Robust 2004, and inconclusive results for gov2 (LOO helped MAP, TAT did not).

One important question is which one of document re-sampling or the use of multiple query variants is responsible for the improved robustness observed in Section 3.8? Second, what is the effect of document resampling alone on precision? The results in Table 3.2 suggest that query variants may be largely account for the improved robustness. When query variants are turned off and the original query is used by itself with document sampling, there is little net change in average precision, a small decrease in P10 for 3 out of the 4 topic sets, but a significant drop in robustness for all topic sets.

3.4.9 The effect of document resampling method

We give two results on document resampling here. First, we examined the effect of two different document sampling methods on retrieval effectiveness. Second, we measured the effect on precision of adding document resampling to the baseline method (which uses no document resampling).

The 'uniform weighting' strategy ignored the relevance scores from the initial retrieval

Collection		DS + QV	DS + No QV
TREC 1&2	MAP	0.2406	0.2547 (+5.86%)
	P10	0.5263	0.5362 (+1.88%)
	RI	0.7087	0.6515 (-0.0572)
TREC 7	MAP	0.2169	0.2200 (+1.43%)
	P10	0.4480	0.4300 (-4.02%)
	RI	0.5652	0.2609 (-0.3043)
TREC 8	MAP	0.2268	0.2257 (-0.49%)
	P10	0.4340	0.4200 (-3.23%)
	RI	0.4545	0.4091 (-0.0454)
wt10g	MAP	0.1946	0.1865 (-4.16%)
	P10	0.2960	0.2680 (-9.46%)
	RI	0.1429	0.0220 (-0.1209)

Table 3.2: Comparison of resampling feedback using document sampling (DS) with (QV) and without (No QV) combining feedback models from multiple query variants.

Collection		QV + Uniform weighting	QV + Relevance-score weighting
TREC 1&2	MAP	0.2545	0.2406 (-5.46%)
	P10	0.5369	0.5263 (-1.97%)
	RI	0.6212	0.7087 (+14.09%)
TREC 7	MAP	0.2174	0.2169 (-0.23%)
	P10	0.4320	0.4480 (+3.70%)
	RI	0.4783	0.5652 (+18.17%)
TREC 8	MAP	0.2267	0.2268 (+0.04%)
	P10	0.4120	0.4340 (+5.34%)
	RI	0.4545	0.4545 (+0.00%)
wt10g	MAP	0.1808	0.1946 (+7.63%)
	P10	0.2680	0.2960 (+10.45%)
	RI	0.0220	0.1099 (+399.5%)

Table 3.3: Comparison of uniform and relevance-weighted document sampling. The percentage change compared to uniform sampling is shown in parentheses. QV indicates that query variants were used in both runs.

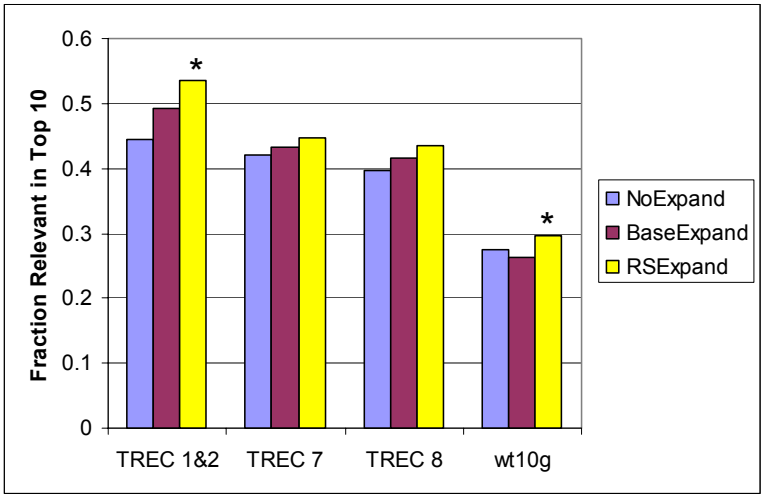


Figure 3.15: The effect of document resampling on baseline expansion P10. The asterisk shows differences that are significant at the 0.05 level.

and gave each document in the top k the same probability of selection. In contrast, the 'relevance-score weighting' strategy chose documents with probability proportional to their relevance scores. In this way, documents that were more highly ranked were more likely to be selected. Results are shown in Table 3.3.

The relevance-score weighting strategy performs better overall, with significantly higher RI and P10 scores on 3 of the 4 topic sets. The difference in average precision between the methods, however, is less marked. This suggests that uniform weighting acts to increase variance in retrieval results: when initial average precision is high, there are many relevant documents in the top k and uniform sampling may give a more representative relevance model than focusing on the highly-ranked items. On the other hand, when initial precision is low, there are few relevant documents in the bottom ranks and uniform sampling mixes in more of the non-relevant documents.

Figure 3.15 shows the differences in P10 for no expansion (NoExpand), the baseline Indri expansion without document resampling (BaseExpand), and baseline expansion with document resampling using relevance-weighted selection (RSExpand), across four TREC collections. We can see that a consistent effect of turning on document resampling across all four collections is to increase the P10 precision of the baseline method. We explore this effect further in Section 3.4.10.

Baseline FB	$p(w_i\|\mathcal{R})$	Resampling FB	$p(w_i\|\mathcal{R})$
said	0.055	court	0.026
court	0.055	pay	0.018
pay	0.034	federal	0.012
but	0.026	education	0.011
employees	0.024	teachers	0.010
their	0.024	employees	0.010
not	0.023	case	0.010
federal	0.021	their	0.009
workers	0.020	appeals	0.008
education	0.020	union	0.007

Table 3.4: Feedback term quality when a stoplist is not used. Feedback terms for TREC topic 60: *merit pay vs seniority*.

3.4.10 The effect of resampling on expansion term quality

Ideally, a retrieval model should not require a stopword list when estimating a model of relevance: a robust statistical model should down-weight stopwords automatically depending on context. Stopwords can harm feedback if selected as feedback terms, because they are typically poor discriminators and waste valuable term slots. In practice, however, because most term selection methods resemble a *tf.idf* type of weighting, terms with low *idf* but very high *tf* can sometimes be selected as expansion term candidates.

This happens, for example, even with the Relevance Model approach that is part of our baseline feedback. To ensure as strong a baseline as possible, we use a stoplist for all experiments reported here. If we turn off the stopword list, however, we obtain results such as those shown in Table 3.4 where four of the top ten baseline feedback terms for TREC topic 60 (said, but, their, not) are stopwords using the Base-FB method. (The top 100 expansion terms were selected to generate this example.)

Indri's method attempts to address the stopword problem by applying an initial step based on Ponte [Ponte 2000] to select less-common terms that have high log-odds of being in the top-ranked documents compared to the whole collection. Nevertheless, this does not overcome the stopword problem completely, especially as the number of feedback terms grows.

Using resampling feedback, however, appears to mitigate the effect of stopwords au-

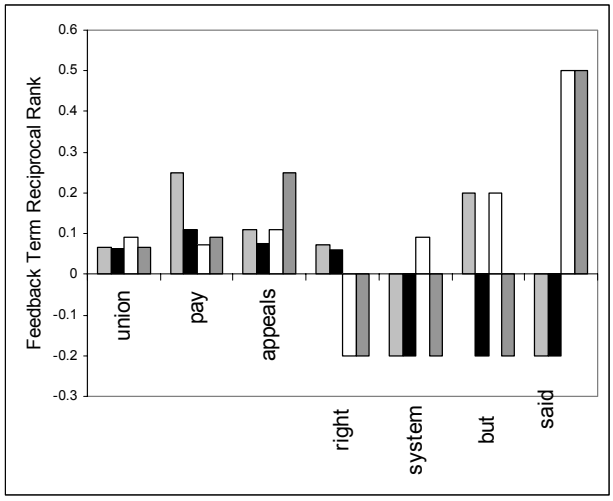


Figure 3.16: Example from TREC topic 60: *merit pay vs. seniority*, showing how term ranking varies as four different bootstrap samples of the top-retrieved documents are used as input to the baseline feedback algorithm. Some terms such as 'union', 'pay' and 'appeals' have a low but stable ranking across all feedback models, while noise words, such as stopwords 'but' and 'said' rank highly in some feedback models, but fail to make the top *m* in others (denoted by a negative rank score). Near-stopwords such as 'right' and 'system' are also shown and are also typically removed because of inconsistent ranking.

tomatically. In the example of Table 3.4, resampling feedback leaves only one stopword (their) in the top ten. We observed similar feedback term behavior across many other topics. The reason for this effect appears to be the interaction of the term selection score with the top-*m* term cutoff. While the presence and even proportion of particular stopwords is fairly stable across different document samples, their relative position in the top-*m* list is not, as sets of documents with varying numbers of better, lower-frequency competing terms are examined for each sample. As a result, while some number of stopwords may appear in each sampled document set, any given stopword tends to fall below the cutoff for multiple samples, leading to its classification as a high-variance, low-weight feature. Thus, as with traditional bagging, the use of resampling for feedback acts to stabilize an unstable relevance predictor for terms.

More insight into this behavior is given by Figure 3.16, which shows how the ranking of seven individual feedback terms varies across four successive bootstrap samples of the top-retrieval documents. A negative rank score on the diagram indicates that the term did not appear in the top-*m* scoring terms for that sample. The first three terms, 'union', 'pay', and

'appeals' all consistently rank in the top-*m* terms chosen by the baseline feedback method, for all four feedback lists. The next two terms, 'right' and 'system' are higher-frequency near-stopwords. 'Right' appears in the top *m* for 2 out 4 samples, while 'system' appears only once out of 4 samples. The stopwords 'but' and 'said' have even higher score variance: when they are selected, they tend to be ranked very high, but they both only appear in 2 out of 4 samples.

The result of this when fitting a Dirichlet distribution to the feedback models is that terms that consistently score highly, and thus are selected for the top *m* terms, are assigned a fairly high α_i parameter, while many stopwords and near-stopwords have much higher score variance and thus receive an α_i closer to 1. The bagging-type behavior becomes clear here; we can think of feedback in terms of a 2-class classification problem identifying 'good' feedback terms vs. 'bad' feedback terms, such that terms scoring in the top *m* are given a label of 'good', and other terms classified as 'bad'. Bagging would then select the terms appearing in a majority of feedback models: in this case, the first three terms shown in the figure.

The reason that high-variance terms are treated as inferior is that we are searching for features that are consistent with multiple samples from a single latent 'relevance' model. We expect the occurrence of such good relevance features to be stable within the set of relevant documents *in competition with other possible term-features*. Such features will maximize the number of relevant documents for which we obtain good discriminators.

3.5 Related work

Our approach is related to previous work from several areas of information retrieval and machine learning. Our use of query variation for feedback was inspired by the work of [YomTov et al. 2005] for query difficulty estimation, who used TAT query variants to generate variance statistics that were used as features for a difficulty classifier. Other related work includes a study Carpineto et al. [Carpineto et al. 2001b], who investigated combining terms from different distributional methods (all based on top-ranked documents from the initial query only) using a term-reranking combination heuristic. In a set of TREC topics they found wide average variation in the rank-distance of terms from different expansion methods.

A study by [Amati et al. 2004] is one of the closest in problem area to our own work: the authors' goal is to obtain more robust expansion using selective expansion methods. Selec-

tive expansion is done by setting a threshold for a heuristic they call *InfoQ*, which combines query length, another heuristic called *InfoPriorQ* which is the normalized deviation of the *idf* values of the query terms, and a factor M_Q which is essentially the normalized deviation of the query clarity score, resulting in the formula

$$InfoQ = \frac{1}{QueryLen} \cdot \left(\frac{InfoPriorQ - \mu_{InfoPriorQ}}{\sigma_{InfoPriorQ}} + M_Q \right). \tag{3.18}$$

Their study had important limitations. First, it was based on only one collection.[11] Second, the authors measured robustness primarily by the increase in the number of topics with no relevant documents in the top 10 (i.e. having P@10 of zero). We believe this is a poor summary statistic for robustness: it completely ignores the magnitude of expansion failure on the many topics that have low but non-zero P@10, and it focuses exclusively on the highest possible P@10 loss (zero), which represent only a fraction of actual expansion failures. The authors also defined $MAP(X)$, the MAP of the X worst-performing topics, but their selective expansion results (p.12) did not use this statistic.

More recently, Crabtree *et al.* developed a system called AbraQ [Crabtree et al. 2007] for automatic query expansion that treats the query as a set of aspects, and aims to find expansion terms that adequately 'cover' all of the aspects in some sense. Because of these shared assumptions we we now give a brief summary of AbraQ and then discuss similarities and differences with our work. The authors evaluated their method on only ten queries using private relevance assessments of Web results from Google, so a direct experimental comparison is not possible.

3.5.1 The AbraQ algorithm

AbraQ first identifies aspects in the original query by scoring word subsequences using their web frequency. The *existence* score of a subsequence $T = w_i \ldots w_j$ is given by

$$Existence(T) = \frac{D_{PHRASE}(T)}{D_{AND}(T)} \tag{3.19}$$

and the *support* score for T is defined as

$$Support(T) = \frac{D_{PHRASE}(T)}{\sum_{T' \in Perm(T) \setminus \{T\}} D_{PHRASE}(T')} \tag{3.20}$$

[11] Robust 2003 topics on TREC disks 4&5 minus Congressional Record.

where $D_{PHRASE}(T)$ is the number of documents containing T as a phrase, and $D_{AND}(T)$ is the number of documents containing all of the terms T without regard to proximity or order. The aspect score $S(T)$ is defined as the heuristic

$$S(T) = Existence(T) \cdot Support(T) \tag{3.21}$$

$$= \frac{D_{PHRASE}(T)}{D_{AND}(T)} \cdot \frac{D_{PHRASE}(T)}{\sum_{T' \in Perm(T)} D_{PHRASE}(T')} \tag{3.22}$$

$$\tag{3.23}$$

where $Perm(T)$ is the set of all permutations of the sequence T. The subsequence of terms T is considered an aspect if $S(T) > 1$. This leads to a set of aspects $\mathcal{A}_Q$ for a query Q where each aspect is defined by a subset of query terms in T.

Once the set $\mathcal{A}_Q$ has been identified, AbraQ calculates a vocabulary model $Vocab(A)$ for each $A \in \mathcal{A}_Q$ using a normalized weighted vector of terms having high co-occurrence with the aspect string A. The model $Vocab(D)$ represents the vocabulary model of the initial top-retrieved documents in response to query Q. AbraQ determines which aspects are underrepresented in the results of the original query Q by computing a similarity-based relative aspect score $RAS(A, D)$ between the aspect vocabulary and the initial document vocabulary vectors, normalized across all aspects.

$$RAS(A, Q) = \frac{Vocab(A) \cdot Vocab(D)}{\sum_{A_i \in \mathcal{A}_Q} Vocab(A_i) \cdot Vocab(D)} \tag{3.24}$$

The *representation level threshold* $RLT(Q)$ is the heuristic

$$RLT(Q) = \frac{1}{1 + |\mathcal{A}_Q|} \tag{3.25}$$

An aspect A is considered valid and underrepresented if

$$0.2 \cdot RLT(Q) < RAS(A, Q) < RLT(Q). \tag{3.26}$$

An aspect is considered invalid if $RAS(A, Q) < 0.2 \cdot RLT(Q)$, in which case further splitting and processing is done to A to search for underrepresented subaspects of A. Any underrepresented aspects are enhanced with additional expansion terms from that aspect's vocabulary model. If AbraQ determines that all aspects are represented adequately, it does

not refine the query.

The AbraQ algorithm requires potentially large numbers of extra queries. According to the authors [Crabtree et al. 2007] for initial queries of n words, in the worst case when each query term is a separate aspect, the number of extra queries is $O(n^2)$. This means that for typical web queries of 2–10 words with aspects of 1–3 words, AbraQ needs between 803 and 4027 count-operations to calculate $D(T)$ and D_{PHRASE} (each of which may be a web query when using the Web as global collection) as well as 53 to 105 additional queries to perform aspect refinement. When typical web query length distributions are taken into account, AbraQ can be expected to use 56 extra queries. In contrast, our methods require dramatically fewer extra query operations: in the worst case, never more than $O(n)$ additional queries on the collection.

3.5.2 Other related work

The idea of examining the overlap between lists of suggested terms has also been used in early query expansion approaches. Xu and Croft's method of Local Context Analysis (LCA) [Xu & Croft 2000] includes a factor in the empirically-derived weighting formula that causes expansion terms to be preferred that have connections to multiple query terms.

On the document side, recent work by Zhou & Croft [Zhou & Croft 2006] explored the idea of adding noise to documents, re-scoring them, and using the stability of the resulting rankings as an estimate of query difficulty. This is related to our use of document sampling to estimate the risk of the feedback model built from the different sets of top-retrieved documents. To be practical, however, their method requires the co-operation of the search engine, whereas our method does not. Sakai et al. [Sakai et al. 2005] proposed an approach to improving the robustness of pseudo-relevance feedback using a method they call *selective sampling*. The essence of their method is that they allow skipping of some top-ranked documents, based on a clustering criterion, in order to select a more varied and novel set of documents later in the ranking for use by a traditional pseudo-feedback method. Their study did not find significant improvements in either robustness (RI) or MAP on their corpora.

Two recent studies have attempted to find good predictors of expansion effectiveness. In his thesis, ([Billerbeck 2005], p.81) made a detailed study of query expansion using a local analysis method, including the effects of varying two parameters: the number of documents in the initial ranking, and the number of expansion terms. He found no correlation between

the average precision of the initial query, and the amount that query was improved by query expansion. He tried several statistics for predicting expansion effectiveness, including a clarity-type query difficulty score, but with no success. [Amati et al. 2004] also reported that 'the performance of query expansion (QE) is not directly related to query difficulty, consistent with the observation that although the retrieval effectiveness of QE in general increases as the query difficulty increases, very easy queries hurt performance'.

Our use of query variants to improve query expansion is related to the recent use of *associated queries*, which are user queries that all share high statistical similarity with a particular document. In [Billerbeck 2005], p.90), sets of associated queries are obtained using Excite query logs. These past queries are then combined into surrogate documents, which are then used for query expansion. With this method, the author's best algorithm obtained a MAP on wt10g of 0.1893. This was the only comparable set of topics with our study, and is in line with the Indri baseline wt10g MAP of 0.1830. For comparison, our RS-FB MAP on wt10g was 0.1984 (about 5% higher). Robustness results were not available, and results for TREC 7 and 8 were apparently poor and not reported. Billerback concludes that associated-based query expansion is most effective when the past queries available are a good match for the collection being searched.

Content-based image retrieval, having to deal with a more varied set of features, has explored sophisticated methods for 'query shifting' – another term for query expansion – based on feedback. The work of [J. Peng 1999] is a good example in this domain, although little or no use of variance estimates or robust model combination methods is performed.

Greiff, Morgan and Ponte [Greiff et al. 2002] explored the role of variance in term weighting. In a series of simulations that simplified the problem to 2-feature documents, they found that average precision degrades as term frequency variance – high noise – increases. Downweighting terms with high variance resulted in improved average precision. This seems in accord with our own findings for individual feedback models.

Estimates of output variance have recently been used for improved text classification. Lee *et al.* [Lee et al. 2006] used query-specific variance estimates of classifier outputs to perform improved model combination. Instead of using sampling, they were able to derive closed-form expressions for classifier variance by assuming base classifiers using simple types of inference networks.

Ando and Zhang proposed a method that they call structural feedback [Ando & Zhang 2005] and showed how to apply it to query expansion for the TREC Genomics Track. They

used r query variations to obtain R different sets S_r of top-ranked documents that have been intersected with the top-ranked documents obtained from the original query q_{orig}. For each S_i, the normalized centroid vector $\hat{w}_i$ of the documents is calculated. Principal component analysis (PCA) is then applied to the $\hat{w}_i$ to obtain the matrix Φ of H left singular vectors ϕ_h that are used to obtain the new, expanded query

$$q_{exp} = q_{orig} + \Phi^T \Phi q_{orig}. \tag{3.27}$$

In the case $H = 1$, we have a single left singular vector ϕ:

$$q_{exp} = q_{orig} + (\phi^T q_{orig})\phi$$

so that the dot product $\phi^T q_{orig}$ is a type of dynamic weight on the expanded query that is based on the similarity of the original query to the expanded query. The use of variance as a feedback model quality measure occurs indirectly through the application of PCA. It would be interesting to study the connections between this approach and our own model-fitting method.

Finally, in language modeling approaches to feedback, Tao and Zhai [Tao & Zhai 2006] describe a method for feedback that allows each document to have a different feedback α. The goal of their method is to make the feedback algorithm less sensitive to the number k of top-ranked documents chosen. The feedback weights are derived automatically using regularized EM. A roughly equal balance of query and expansion model is implied by their EM stopping condition. They propose tailoring the stopping parameter η based on a function of some quality measure of feedback documents.

There has already been substantial work in the field of machine learning on ensemble methods that resample the training data in various ways. In this respect, our approach resembles methods such as *bagging* [Breiman 1996], an ensemble approach which generates multiple versions of a predictor by making bootstrap copies of the training set, and then combines the predictions of the base classifier by averaging (for numerical predictors) or majority vote. Bagging has been proven effective at stabilizing the performance of unstable base classifiers. In our application, top-retrieved documents can be seen as a kind of noisy training set for relevance. Thus, by viewing retrieval functions such as pseudo-relevance feedback as learning algorithms trained using sampling-based methods, we can

apply a broad body of existing results and methods previously applied to tasks such as text classification.

3.6 Computational complexity

Running extra query variants and creating multiple feedback models provide us with valuable information about query and document uncertainty, but they also have extra computational costs. We now explain the nature of these costs and how they can be mitigated. Our focus in this chapter has been on finding which methods work best, and not on optimizing performance.

In the case of document set resampling, we have two main steps:

1. Estimating B feedback models
2. Fitting the Dirichlet distribution to these models.

For Step 1, the k document representations must be loaded into memory. This happens whether we do standard pseudo-relevance feedback, or resampling feedback. For each of the B feedback models, the costs are roughly as follows. With k documents each on average λ words long, from Heaps Law for English we have a vocabulary size of roughly $d = 45 \cdot \sqrt{\lambda k}$. For example, when $\lambda = 1000$ and $k = 50$, we have a vocabulary vector size of about 10000 unique words. We perform $K \cdot d$ floating point calculations to obtain term weights, where K is a small constant[12], including a sort of $O(d \log d)$ to find the top m feedback terms. Step 1 is an ideal case for parallel implementation if we care to do so: all B models can be computed in parallel and can all share the same data block.

For Step 2, the computation costs are T_α iterations for finding the α parameters and T_s iterations for finding the scale parameters. In our code, $T_\alpha = 5$, $T_s = 3$. For either α or scale parameter fitting, each iteration requires about $d \cdot B$ calls to the log function and $2d$ calls to digamma-family functions[13]. In real numbers, to perform step 2 for a query amounts to about $10,000 \cdot 10 \cdot (5 + 3) = 800,000$ calls to log and $2d \cdot (5 + 3) = 160,000$ calls to a digamma-family function. While these counts might seem high, we note that on an Intel Xeon CPU (3.20 GHz) using the standard C++ math library, the above calls took a total of 0.17 seconds. In addition, our numerical code has not been extensively tuned for speed, so that further performance improvements may be possible.

[12] K is typically less than 10

[13] In addition to a small number of $O(d)$ steps such as finding the mean of d values

In the case of query variants, the computational cost is somewhat higher. For example, the Indri search engine constructs an inference network based on the user's query. However, the nature of our basic query variation methods is such that only the *weights* of term evidence nodes in the network are changed. Except for the weight values, the inference network structure itself is identical for all variants. Potentially the most expensive operation in constructing the network is fetching and decompressing the inverted list from disk for each term. However, because the same inference network can be used in all cases, this cost is amortized across all query variants. Furthermore, the internal design of the Indri search engine is such that scoring a document against multiple term-weight variants is possible to do simultaneously and efficiently [Strohman 2007]. Thus, if embedded into the core search algorithm, the marginal increase in computation cost is very low.

Even if query variation originates completely external to the search engine, multiple query variants submitted close in time are still likely to be processed relatively efficiently: IR system caching or operating system paging will potentially reduce the large inverted list costs via caching. Multiple query variants can be processed in parallel, reducing the effect on user response time (but of course, increasing total CPU load proportionally). Finally, the number of subqueries could easily be adapted to the load on a server: under high load, the strategy would simply degrade toward the performance of the original query.

3.7 Discussion

We now comment on connections between our approach, bagging, and Relevance Models, followed by some possibilities for future extensions.

3.7.1 Connections to bagging

Unlike traditional bagging, the output from our 'classifier', the baseline feedback method Φ, is not a single numeric or categorical response, but a language model represented by a high-dimensional vector. Thus, we do not simply average the vectors that result from resampling the top-retrieved documents. Instead, we solve a slightly more general and appropriate problem and find the maximum likelihood solution to a latent Dirichlet distribution fitting the observed multinomial feedback models. We have observed that simple averaging typically gives significantly worse average precision than using the mode of the fitted Dirichlet distribution.

3.7.2 Connections to the Relevance model

The use of a kernel to smooth document model estimates in the Relevance Model [Lavrenko 2004] creates an effect similar to bootstrap sampling. By perturbing the underlying document model estimate (according to its treatment as a random variable with the kernel distribution), we induce a perturbation in the document's query likelihood score. Since documents are weighted by this score in calculating their contribution to the feedback model, the kernel effectively acts to smooth document weights. The bootstrap method seeks to approximate/simulate this effect by sampling the relative document weights directly from some hypothesized distribution, where the observed document weights are treated as the most likely draw from that distribution. In his thesis, Lavrenko did not evaluate the use of more sophisticated document kernels such as a Dirichlet kernel in an actual retrieval setting, due to its computation burden. However, based on our analysis, we believe the effect of such document kernels is expected to be similar to the bagging-type effect of bootstrap sampling and thus a precision-enhancing device.

3.7.3 Future extensions

The following query variant methods were not implemented for this study, but we mention them for possible future investigation.

- *Document-based variants* (DV). To capture more realistic covariance structure, we can consider a method by Bennett [Bennett 2007] created to estimate the sensitivity of text classifiers. Query models are sampled deterministically from a Voronoi cell around the original query. This cell is created from modified query models, each of which is an interpolation of the original query model with some top-retrieved document. The interpolation parameter α_i for each modified query $\boldsymbol{q}_i$ is set to the largest value that keeps the original query as the nearest neighbor to the modified model. Since each α_i requires several retrieval operations to estimate, the computational overhead for this method would likely be several times greater than the most sophisticated method used in this thesis, sigma-point sampling.

- *Translation models* (TM). These use kernels defined on graphs of term relations to define a translation mapping between term w_i and term w_j. An example of this approach was recently applied to text classification by [Dillon et al. 2007]. A related

approach using term dependencies for query expansion was studied by [Collins-Thompson & Callan 2005]. The *perturbation kernels* that we learn in Chapter 4 from query variants may also be seen as a form of translation model trained from local (top document) data, and we evaluate the effect on query expansion performance in Section 4.3.3.

Different document sampling strategies are also possible. The following methods were not implemented for this study but are candidates for testing.

- *Dirichlet over weights.* This would model the relative weights of documents with a Dirichlet distribution.
- *Logistic normal.* Extending the idea of a document as a topic, similarity between document topics could be modeled using a covariance matrix. The logistic normal distribution described later in Section 4.2.4 could easily model more complex covariance structure.

3.8 Conclusions

We have presented a new approach to pseudo-relevance feedback based on document and query sampling. While our study uses the language modeling approach as a framework for experiments, we make few assumptions about the actual workings of the feedback algorithm. Our results on standard TREC collections show that our framework improves the robustness of a strong baseline feedback method across a variety of collections, without sacrificing average precision. It also gives small but consistent gains in top-10 precision, which is typically difficult to do. In general, our approach has the following advantages.

First, the framework is *flexible*: it assumes little about the baseline feedback method. We observed consistent improvements in precision and robustness using different strong baseline feedback algorithms. We believe it is likely, therefore, that any baseline algorithm obtaining good initial performance, especially if unstable with respect to small changes in the top-retrieved documents, will benefit from our method.

Second, the sampling approach is *powerful*, because it gives a principled way to get variance or sensitivity estimates from any black box process, allowing us to quantify the uncertainty associated with important random variables such as document or query model parameters. We showed that using such variance calculations resulted in more effective

model combination of different feedback models. We also showed that the confidence estimates obtained for sets of expansion terms may be used to perform selective expansion. Furthermore, we discussed how sampling methods can extend the range of retrieval scoring functions for which current Bayesian retrieval frameworks are practical.

Third, the algorithms here are *generic*: extensive parameter training is not essential to achieve good performance. This does not mean that further improvements are not worth pursuing with parameter tuning based on training data. For example, inductive transfer may certainly be possible to perform meta-learning across different retrieval problems or corpora. For standard use, however, the effective values of important parameters such as α (feedback interpolation), k (number of top-ranked documents), etc. appear stable across a variety of collections.

Fourth, our method is *precision-oriented*, consistently increasing the percentage of relevant documents in the top 10 retrieved documents. This is a desirable quality for applications such as web search where accuracy in the first few results is desirable. We showed that this increase in precision is likely due to the bagging-like effect of emphasizing feedback terms that have low variance across multiple samples, i.e. those that are consistent across related hypotheses of relevance.

Fifth, resampling is *stable* and provides more robust feedback. We showed how query-side resampling is responsible for most gains in robustness, by emphasizing terms that are consistently related to multiple query aspects. We also analyzed the trade-off between robustness and average precision, and showed that when query and top-document resampling were used together, the resulting feedback model gave a tradeoff curve significantly better than the baseline language model, with better tradeoff values for a wide range of the interpolation parameter α.

As with any ensemble-type approach that relies on obtaining multiple sources of evidence, there is increased computational cost. In Section 3.6 we summarized how to mitigate these costs, using Indri [Strohman et al. 2004] as a specific example. We believe the gain in retrieval quality and increased family of practical scoring functions that sampling makes possible are an acceptable tradeoff.

The next chapter demonstrates an additional benefit to running query variants that further amortizes their cost: exactly the same result information can be re-used for learning improved query-specific similarity measures or query difficulty measures.

Chapter 4

Data Perturbation Kernels

In this chapter we introduce another novel application of sampling to information retrieval: learning query-specific similarity measures using small perturbations to the original query. We will use these similarity measures in Chapter 6 to model term relationships, which will result in more reliable model combination results than the heuristic methods we have just described in Chapter 3. Therefore, we digress for one chapter to focus on effective term similarity measures.

We call the general class of kernels that we introduce for this purpose *data perturbation kernels*, because they are learned from the results of a small number of perturbations to the training data. For example, when the training data is a query, a perturbation may be a small change in the relative weighting of the original terms. In this way we obtain new information about the *changes* to feedback models (or other statistical models) in a neighborhood of the original query. Our approach differs from existing kernel families such as probability product kernels that assign exactly one probability density to each point in the input space $\mathcal{X}$ and then integrate over $\mathcal{X}$. Instead, we take essentially the dual view and identify each input point with multiple probability densities evaluated at that point, and then integrate over probability density space. The resulting algorithms are generally more effective than similarity measures based on the original query alone, have a principled mathematical foundation, are relatively simple to implement, and can be extended to arbitrary retrieval objects.

The key idea of data perturbation kernels is that two input objects x and y, such as terms or language models, are considered similar in the context of a given query Q if the probability distributions $p(x|Q)$ and $p(y|Q)$ that depend on Q are affected in similar ways with

small variations in Q. For example, the probability of terms x and y may covary in generative relevance models $p(\cdot|R)$ estimated using variants of Q. We call this the *perturbation similarity* of x and y.

As we show below, the 'ideal' perturbation similarity measure (PSM) with respect to Q takes the form of an integral over all possible queries in a neighborhood of Q. Computing the exact integral is not possible since we do not know the 'true' distribution of queries given the unobservable information need. However, we can apply general-purpose integration methods such as importance sampling (described in Chapter 2) to learn an approximation to the true PSM by using query perturbations (sometimes referred to as *query variants*). Since there is a computational cost to searching with query variants, and many potential methods to generate variants, we develop algorithms for choosing a small set of 'good' query variants. In particular, we give an efficient deterministic algorithm called *sigma-point sampling* that is effective for arbitrary integrands in the perturbation similarity integral. Sigma-point sampling can also be thought of as a general query variant generation method that behaves like an adjustable combination of LOO and TAT sampling described in Chapter 3.

Our motivation for introducing data perturbation kernels is to create higher-quality *query-specific* similarity measures. For example, a problem with many existing word similarity measures is that they do not adequately handle *polysemy*, a condition in which a word has multiple meanings or *senses*. These senses can be very different: the word *wave* can refer to a water phenomenon, a hand motion, and so on. Typical similarity measures ignore the *context* in which two words are used, and focus only on some function of their joint appearance or non-appearance. When context is ignored, the related words for different senses of the same word are incorrectly calculated to be 'close' to each other, because each is close to the original word. Unlike static similarity measures, such as word co-occurrence in a general collection, query-specific similarity measures use the *other query terms* to provide additional context for the similarity computation.

A specific query example occurs with the word "java", which has multiple meanings: it can refer to a drink (coffee), a programming language, or a place. Many word similarity measures would give all three related words "coffee", "programming", and "indonesia" strong similarity scores to "java", because such measures rely on general statistics that ignore context, such as word co-occurrence in a large, generic corpus. Knowing that "java" is used in a query like "java interpreter download", however, the perturbation kernel automat-

ically discounts similarity between words that that do not also have significant connections to the query terms. In this case, the words “coffee” and ”indonesia” do not exhibit strong connections to ”interpreter” and ”download” and thus would be considered much less related to ”java” than words like ”programming” that reflect the correct sense.

The rest of this chapter is organized as follows. Section 4.1 gives an overview of the problem and our solution. In Section 4.2 we give a detailed mathematical derivation of our similarity formulas. We then give two applications of our approach. First in Section 4.3 we derive a new similarity measure between words called *perturbation similarity* for clustering terms for query expansion. Second, in Section 4.4 we apply the same kernel in a different domain – that of language models – and use this to obtain a *generalized clarity score* that extends the existing method of the same name by Cronen-Townsend and Croft [Cronen-Townsend & Croft 2002]. In Section 4.5 we confirm the effectiveness of perturbation similarity, especially for collections in which relevant documents are highly clustered. We discuss connections to related work in Section 4.6. Section 4.7 summarizes our conclusions. In Appendix B we discuss related work on measures of *local influence* in statistics that are closely connected with the notion of perturbing the query as training data.

4.1 Overview

In Chapter 3, we developed the idea of using query variants to improve the performance of query expansion. Each query variant $\boldsymbol{q}_i$ resulted in a new set of retrieved documents $\mathcal{D}_i$ and a corresponding set of feedback models $\{\theta_i\}$. Here, we are interested not in combining the models, but extracting information from them about term similarity, which we can then use for more effective model combination. In machine learning terms, the use of query variants to derive a similarity measure between words can be considered a form of multi-task metric learning in which each query variant is an auxiliary task. Two terms u and v can be compared by how their probabilities $p(u|\theta_i)$ and $p(v|\theta_i)$ covary across all models $\{\theta_i\}$. These *perturbation kernels* would be expected to be more precise because they retain the query as critical context, as opposed to similarity learned from a general large corpus. Such kernels could be applied in many domains of interest, and here we focus on two important domains: terms, and language models.

We next give the complete mathematical formulation of data perturbation kernels, proceeding in three steps. First, after introducing basic concepts, we give the ideal but computationally intractable version of a basic similarity function in terms of an integral over

the space of query functions. In this formulation, a query *neighbourhood* specifies a probability distribution around the original query, which is treated as a function. In this view, each query variant is a point in function space near the original query, and the query neighborhood then forms the probability measure for the similarity integral. Finally, we give an algorithm for an efficient approximation of the similarity integral that requires only a small number of deterministic samples in query function space (query variants). As we did in Chapter 3, we model the query $\boldsymbol{q}$ as generated by a latent mixture model with each mixture component corresponding to a different hypothesis about the information need. Each hypothesis corresponds to a different query variant which in turn is used to estimate a different generative model θ_i of relevance. We choose a latent model such that the maximum likelihood model is the one derived from the original query, $\theta_{\boldsymbol{q}}$. We then compute a distance or loss function of each variant's model parameter(s) θ_i relative to the maximum likelihood parameters of $\theta_{\boldsymbol{q}}$.

4.2 Mathematical formulation

Our goal in this section is to derive a rigorous mathematical method for estimating query-specific similarity, based on treating the query string as training data. To achieve this goal, we solve a more general problem by introducing a data-dependent kernel called the *data perturbation kernel*. The data perturbation kernel compares the nature of the *sensitivity* of the probability estimates for two input points x and x' to perturbations of the training set. As such, it focuses on relative change in probability, and not absolute differences.

4.2.1 Basic concepts

We first introduce some notation and basic concepts. Let $\mathcal{X}$ be the input domain from which training examples $\boldsymbol{x}_i$ are drawn. The set $\mathcal{X}$ may be a finite set, as with a vocabulary $\mathcal{V}$ of words, or infinite as with a real vector space $\mathbb{R}^d$. Let $\mathcal{P}$ be the set of probability distributions on $\mathcal{X}$. A *training set* $\boldsymbol{x}$ of examples is drawn from $\mathcal{X}$ and a density estimate $p(\boldsymbol{x}) \in \mathcal{P}$ is created from the examples using an *estimator* function $\mathcal{R} : \mathcal{X}^n \rightarrow \mathcal{P}$. Note that $p(\boldsymbol{x})$ may be either parameteric or non-parametric. For example, $\mathcal{R}$ may be a complex algorithm that estimates a Relevance Model density $p(w|\theta)$ over a finite vocabulary space of words $\mathcal{X} = \mathcal{V}$. (This would be a parametric model with parameters θ.)

Feature mapping and kernels. For any point $\boldsymbol{x}$ in input space $\mathcal{X}$ we identify an m-dimensional vector called a *feature mapping*, denoted $\phi : \mathcal{X} \rightarrow \mathbb{R}^m$. A feature mapping

is a different representation of $\boldsymbol{x}$ in the feature space $\mathcal{F} = \mathbb{R}^m$. Note that $\mathcal{F}$ may be either finite- or infinite-dimensional depending on the feature space. Given a feature mapping $\phi(x)$, we define a symmetric function $k(\cdot)$ to measure the closeness of input points x and y:

$$k(x, y) = \phi(x) \cdot \phi(y) \tag{4.1}$$

where the right side of the equation is the *inner product* of $\phi(x)$ and $\phi(y)$ in the space $\mathcal{F}$. If we have a set of n objects $\boldsymbol{x} = \{x_i\} \in \mathcal{X}$ then the matrix $\boldsymbol{K}_x$ with i, j entry $k(x_i, x_j)$ is called the *kernel matrix* for the set $\boldsymbol{x}$. The function $k(\cdot)$ is called a *kernel function* if the kernel matrix is always positive semi-definite for any subset of objects from $\mathcal{X}$. Given two instances $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from $\mathcal{X}$, the 2-norm *distance* $\delta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in terms of the kernel k is given by

$$\delta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|_2 \tag{4.2}$$

$$= \sqrt{k^2(\boldsymbol{x}_i, \boldsymbol{x}_i) + k^2(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \tag{4.3}$$

Perturbations to input data. A *perturbation* to a training set of n instances $\boldsymbol{x} = \{\boldsymbol{x}_1 \ldots \boldsymbol{x}_n\}$ can modeled by a vector of counts $\boldsymbol{\alpha} = \{\alpha_1 \ldots \alpha_n\}$ with count α_i corresponding to the weight of training example $\boldsymbol{x}_i$. For the original training set, $\alpha_i = 1$ for all instances $\boldsymbol{x}_i$. To leave out the instance $\boldsymbol{x}_i$, we set $\alpha_i = 0$. To give $\boldsymbol{x}_i$ more weight, we set $\alpha_i > 1$. A *perturbation strategy* is a set $\mathcal{A}$ of perturbation vectors. The set $\mathcal{A}$ may be selected with either a random or deterministic process. Two examples are:

- The *leave-one-out* strategy is deterministic, with $\mathcal{A} = \{\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_m\}$ where $\boldsymbol{\alpha}_i[j] = 0$ for $i = j$ and 1 otherwise.

- The *Dirichlet* strategy randomly samples m vectors using the Dirichlet $Dir(\mathbf{1}^n)$.

It will be convenient to denote the probability density of a variable $x \in \mathcal{X}$ that results from a perturbation α_i to the training set $\boldsymbol{x}$ as $p_{(i)}(x)$.

In the context of information retrieval, we treat the query $\boldsymbol{q}$ as training data for relevance. If we take as our domain $\mathcal{X}$ the set of all possible words $\mathcal{V}$ we can view $\boldsymbol{q}$ as consisting of the unordered set of terms $q_1 \ldots q_n$, and each term q_i may be thought of as an item of training data selected from $\mathcal{X}$, so that $\boldsymbol{q} \in \mathcal{X}^n$. Note that we have already processed

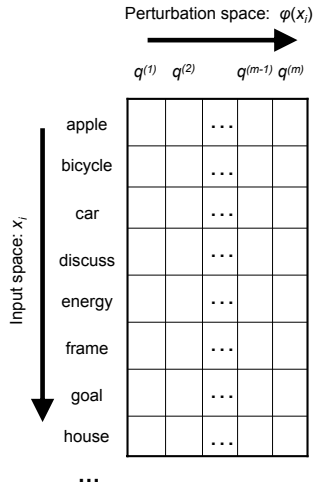


Figure 4.1: The matrix of probability vectors for a discrete input space (here representing a word vocabulary). Each column represents the discrete parameteric or non-parametric probability distribution across all words estimated from a particular perturbation of the training data (i.e. query). Conversely, the rows $\phi(\boldsymbol{x}_i)$ give the probability estimates across all perturbations for a given word $\boldsymbol{x}_i$.

the same query variants for robust feedback as described in Chapter 3, so we essentially get the use of the resulting feedback models 'for free' to use in estimating the kernel $k(\cdot)$.

Figure 4.1 shows the matrix A that results from using m training set perturbations with an example vocabulary. One way we might define ϕ is

$$\phi_k(\boldsymbol{x}_i) = A_{ik} = p(\boldsymbol{x}_i|q_{(\alpha_k)}) \tag{4.4}$$

where A_{ik} is the (i,k)-th entry of A. In this case, the i-th column of A is just the probability distribution over the vocabulary $\mathcal{X}$ estimated from the training set (query) perturbation $\boldsymbol{q}_i$. The row vector $\phi(\boldsymbol{x}_i)$ is the feature mapping for the word $\boldsymbol{x}_i$. A simple form of perturbation similarity $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is simply the dot product between rows:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{m} A_{ik}A_{jk}. \tag{4.5}$$

It is important to note, however, that there are many choices for how to define ϕ, and thus A. For example, we may want to define ϕ such that the dot product of two column vectors corresponds to a particular metric between probability densities. In the next section,

we generalize this example to similarity of two objects in an arbitrary domain, so that the simple sum of Eq. 4.5 becomes an integral expressing a general inner product over some feature space.

4.2.2 Canonical similarity integrals

Our formulation of perturbation similarity is inspired by earlier work of Baxter using auxiliary tasks for classification and function approximation. Baxter showed that for 1-nearest-neighbor classification, there is a unique optimal similarity measure that he called the Canonical Distortion Measure (CDM) [Baxter 1997]. In the classification setting, this quantity $\delta(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is the expected loss of classifying $\boldsymbol{x}_1$ with $\boldsymbol{x}_2$'s label. The expectation is taken over a probability space of classifiers (tasks). The CDM for a particular type of task is uniquely defined by two factors: the choice of loss function and the task distribution. Note that the CDM may not be symmetric or satisfy the triangle inequality, so it does not technically define a metric.

In this chapter we apply Baxter's idea to information retrieval applications, where we view a task as relevance estimation with respect to a particular query $\boldsymbol{q}$. We call a task distribution for $\boldsymbol{q}$ a *query neighborhood* of $\boldsymbol{q}$.[1] We define the query neighborhood as a probability measure $Q(f)$ over the space of query functions $\mathcal{F}$. The canonical similarity measure $\Delta_{\boldsymbol{q}}(x, y)$ is the expected loss over Q, given x and y in the input domain $\mathcal{X}$.

$$\Delta_{\boldsymbol{q}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{\mathcal{F}} \rho(f(x), f(y)) dQ(f) \tag{4.6}$$

This measure is uniquely determined by the task function f, the choice of query neighborhood measure Q and loss function $\rho(u, v)$. We focus on the squared-loss function $\rho(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^2$. This gives the distance measure $A(\theta_1, \theta_2)$:

$$A_Q(\theta_1, \theta_2) = \int_{\mathcal{F}} \rho(f(\theta_1), f(\theta_2)) dQ(f) \tag{4.7}$$

$$= \mathbb{E}_Q[(f(\theta_1) - f(\theta_2))^2] \tag{4.8}$$

$$= \mathbb{E}_Q[f(\theta_1)^2] + \mathbb{E}_Q[f(\theta_2)^2] - 2\mathbb{E}_Q[f(\theta_1) f(\theta_2)] \tag{4.9}$$

[1] In Baxter's terminology this would be called a *query environment*.

where the expectation is taken over the query neighborhood Q.[2]

As Figure 4.1 suggests, there are two approaches to computing similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in $\mathcal{X}$, corresponding to comparing rows or columns in matrix A.

First, we can pick some mapping that identifies $\boldsymbol{x}_i$ with a single *column* vector of A, representing a probability distribution $p_i(x)$ defined over all of $\mathcal{X}$, and then compute a similarity or inner product between distributions $p_i(x), p_j(x)$. We call this *integration over input space*, because that is the domain for the inner product over distributions.

Alternatively, we can identify $\boldsymbol{x}_i$ with the *row* vector of features $\phi(\boldsymbol{x}_i)$. Each entry $\phi_k(\boldsymbol{x}_i)$ corresponds to the result of using k-th query perturbation as training data. This is essentially Baxter's measure and in the limit integrates over the continuous domain of tasks (queries). We call this *integration over perturbation space*.

Integration over input space When we assign each input point $\boldsymbol{x}_i$ a probability distribution over input space, we can integrate over input space – in the discrete case, the *columns* of A. This type of similarity measure has been the subject of recent work by other authors. Jebara, Kondor and Howard [Jebara et al. 2004] introduced *probability product kernels* (PPK) where the kernel function is

$$k_{PPK}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \int_{\mathcal{X}} p_i(x)^{\beta} p_j(x)^{\beta} dx \tag{4.11}$$

for a positive constant β.

A related special case when each input point $\boldsymbol{x}_i$ is identified with its own perturbation $\boldsymbol{q}^{(i)}$ leaving out $\boldsymbol{x}_i$ results in the *leave-one-out* (LOO) kernel [Tsuda & Kawanabe 2002], which measures the similarity of two domain members according to the similarity of their influences on the density when the samples are left out of a given training set. The LOO kernel works for both parametric and non-parametric densities, by measuring the distance between arbitrary probability distributions using the Hellinger norm [Tsuda & Kawanabe 2002].

$$k_{LOO}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 4(n-1)^2 \int (\sqrt{\hat{p}^{(i)}}(\boldsymbol{x}) - \sqrt{\hat{p}}(\boldsymbol{x}))(\sqrt{\hat{p}^{(j)}}(\boldsymbol{x}) - \sqrt{\hat{p}}(\boldsymbol{x}))d\boldsymbol{x} \tag{4.12}$$

[2]Other loss functions, such as the Hamming metric

$$\rho(x_1, x_2) = 1 - \delta(x_1, x_2) \tag{4.10}$$

are also possible.

Tsuda and Kawanabe showed that for parametric probability densities the Fisher kernel [Jaakkola et al. 1999] is a close approximation to the LOO kernel and converges to the LOO kernel as the number of training samples n goes to infinity.

An importance sampling approximation to the LOO kernel can be derived by taking $\hat{p}(\boldsymbol{x})$ as the sampling distribution, with the $\boldsymbol{x}_i$ as the evaluation points. This gives

$$k_{LOO}(x_i, x_j) \approx \frac{4(n-1)^2}{n} \sum_{i=1}^{n} \frac{(\sqrt{\hat{p}^{(i)}}(\boldsymbol{x}_i) - \sqrt{\hat{p}}(\boldsymbol{x}_i))(\sqrt{\hat{p}^{(j)}}(\boldsymbol{x}_i) - \sqrt{\hat{p}}(\boldsymbol{x}_i))}{\hat{p}(\boldsymbol{x}_i)} \tag{4.13}$$

which in turn can be written in the form

$$k_{LOO}(x_i, x_j) \approx \sum_{i=1}^{n} \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) \tag{4.14}$$

Thus, by defining the matrix A with n columns (one for each leave-one-out distribution) and setting the column vectors to have the values

$$\phi_i(\boldsymbol{x}) = \frac{2(n-1)}{\sqrt{n}} \frac{\sqrt{\hat{p}^{(i)}}(\boldsymbol{x}) - \sqrt{\hat{p}(\boldsymbol{x})}}{\sqrt{\hat{p}(\boldsymbol{x})}} \tag{4.15}$$

then finding a good approximation to the leave-one-out kernel reduces to computing a simple relative change in (square-root transformed) probability, and taking the dot product of the resulting columns.

One limitation of the LOO-type kernel is that it is only defined for elements (e.g. vocabulary) that actually exist in the training set. For domains like information retrieval where the training data (query) is extremely limited, this is a significant problem, since it implies that the LOO kernel is only available between those words that happen to occur in the query. This is clearly not acceptable if we wish to perform any kind of useful term clustering.

Integration over perturbation space If we take inner products between the perturbation *rows* in matrix A, this gives the data perturbation kernel that we introduce in this chapter. In contrast to the probability product family of kernels, integration over perturbations does not assume a mapping from each point $\boldsymbol{x}_i$ to a specific density $p_i(\boldsymbol{x})$.

Instead, we fix the two input domain elements $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and integrate over a probability space $\mathcal{P}$ of density functions. These density functions $p_{(\alpha)}(\cdot)$ are those that result from perturbations α on the training data, and we assume we have a measure $Q(p)$ over $\mathcal{P}$ that

describes the distribution over perturbation densities $p_{(\alpha)}(\cdot)$. We assume that the density $p_{(\alpha)}(\cdot)$ is defined for all elements of the input domain $\mathcal{X}$ (although it may be zero), so that the integral exists for all pairs $(x, y) \in \mathcal{X}^2$, and is

$$\Delta_{\boldsymbol{q}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{\mathcal{P}} \rho(p(\boldsymbol{x}_1), p(\boldsymbol{x}_2)) dQ(p) \tag{4.16}$$

Note that this has close connections to the scenario corresponding to the Canonical Distortion Measure, in which each density function $p_{(\alpha)}(\cdot)$ corresponds to a auxiliary prediction task for the main prediction task $p_{\boldsymbol{q}}(\cdot)$ using original query $\boldsymbol{q}$.

In the special case where we use the loss function $\rho(u, v) = u^\beta v^\beta$ for some constant β, we get the counterpart to the probability product kernel in the probability density domain (instead of the input domain).

$$\Delta_{\boldsymbol{q}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{\mathcal{P}} p(\boldsymbol{x}_1)^\beta p(\boldsymbol{x}_2)^\beta dQ(p) \tag{4.17}$$

4.2.3 Approximating the similarity integral

By writing the similarity measure as an integral in Eq. 4.6. we can now bring to bear the general-purpose integration methods described in Chapter 2.

Our approach is to formulate a Monte-Carlo-like estimate by sampling density functions from the domain $\mathcal{P}$. Since each sample corresponds to a query variant, the question then becomes how to choose specific query variants. Our method of selecting samples will be a form of *importance sampling*, using knowledge of the properties of $Q(p)$.

Recall that the basic approach to evaluate a general integral of the form

$$\mathcal{I} = \int_{\Theta} f(\theta) d\mu(\theta) \tag{4.18}$$

on the domain Θ with measure $d\mu$ is to independently sample N points $X_1, \ldots X_N$ in Θ according to some density function $p(x)$, and then compute the random variable

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{4.19}$$

This approach is very general and ignores any information we have about the nature of

f. If we know that f is concentrated in particular areas, we can obtain a more efficient, lower-variance estimate by using *importance sampling* of f.

Suppose $p_{\boldsymbol{q}} \in \mathcal{P}$ is the density function over $\mathcal{X}$ associated with using the original query $\boldsymbol{q}$ as training data. Let $p_{(\alpha)}$ be the density function resulting from training on a perturbation α with $\boldsymbol{q}$. Then informally when $\alpha \approx \mathbf{1}^{|\boldsymbol{q}|}$ – i.e. a 'small' adjustment to $\boldsymbol{q} - p_{(\alpha)}$ will typically be close to $p_{\boldsymbol{q}}$, with the measure $Q(p)$ concentrated there. Thus, importance sampling in the region of $p_{\boldsymbol{q}}$ is likely to be an effective strategy. We now elaborate on specific choices of importance sampling strategy over the query neighborhood probability measure $Q(p)$ to evaluate the integral in Eq. 4.17.

4.2.4 Importance sampling with query neighborhoods

Our goal in this section is to define the PSM for a given query by defining the probability measure $Q(p)$ over the space of queries $\mathcal{F}$. We define a *query neighborhood* density function $h(\theta)$ for a query $\boldsymbol{q}$ that describes how the query model behaves as a random variable with respect to the original query. One way to interpret this is how different query models for the same underlying information need might be generated. In essence, by defining $Q(p)$, which we call the *neighborhood* of $\boldsymbol{q}$, we are stating our assumptions about how we characterize the uncertainty inherent in $\boldsymbol{q}$. The result is that, instead of keeping $\boldsymbol{q}$ fixed as in traditional systems, we treat it as a random variable. Using the distribution $Q(f)$, we will create N query variants $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$.

We now discuss parametric and non-parametric methods for specifying $h(\theta)$. The simplest strategy is to choose no variants at all, and simply use the original query. In this case, this corresponds to choosing a measure $Q(f)$ so that the original query function $f_{\boldsymbol{q}}$ is the unique mode of the distribution, and the integral is estimated using a single sample at the mode:

$$\Delta_{\boldsymbol{q}}(x, y) \approx \rho(f_{\boldsymbol{q}}(x), f_{\boldsymbol{q}}(y)) \tag{4.20}$$

We can also use the term-at-a-time (TAT) and leave-one-out (LOO) methods we used in Chapter 3. The advantage of the TAT and LOO methods is that they are extremely simple to implement. They both use $N + 1$ variants of the query, while being somewhat complementary strategies, making them ideal for comparison. The disadvantage of TAT is that, since only single terms are used, it may not adequately capture covariation between pairs of terms, especially as the query length increases. As we showed for query expansion in Section 3.4.8, and will show in Section 4.5, this leads to poor overall performance on

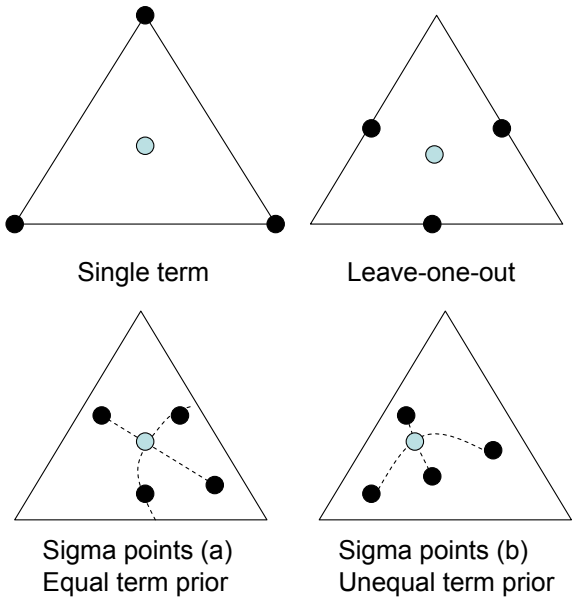


Figure 4.2: Examples of different query sampling strategies, visualized on the simplex of term weights for a three-term query. The original query, with all term weights equal, is in the center of each simplex, except for sigma points (b), where its location is determined by term-specific prior parameters. The black dots denote query variants that result from each scheme. Shown are term-at-a-time (TAT, top left), leave-one-out (LOO, top right) and the sigma points method (second row) for uniform term prior (bottom left) and non-uniform priors (bottom right).

various tasks.

We now define a family of parametric query neighborhoods that behaves somewhat like an adjustable, hybrid form of both TAT and LOO.

Parametric query neighborhoods with the logistic normal distribution

If we have a vocabulary of $d+1$ terms, we may have a unigram model over this vocabulary, in which case it is convenient for us to work in the d-dimensional simplex. However, the unscented transform that we describe in the next section is based on a Gaussian distribution in unbounded Euclidean space. We tie together these two spaces by defining $h(\theta)$ to be a flexible parametric density based on the *logistic normal* distribution [Aitchison & Shen 1980] over the simplex of term weights. The logistic normal distribution provides a simple way to embed a Gaussian input distribution into the d-dimensional simplex. As a special case, when the query neighborhood is well-approximated using a parameteric distribution called the logistic normal, we produce an effective set of query variants using the unscented transform from particle filtering.

The logistic normal implies the following generative process for a set of query term weights. First we draw a set of latent variables $\eta \sim (\mu, \mathbf{\Sigma})$ where $N(\mu, \Sigma)$ is a $k-1$-dimensional Gaussian distribution. Then we project $\boldsymbol{\eta}$ to the k-dimensional simplex point $\boldsymbol{r}$, representing the relative query term weights, with j-coordinate r_j using the logistic transformation

$$r_j = \frac{\exp \eta_j}{\sum_{i=1}^{k} \exp \eta_i}$$

While the Dirichlet distribution can only capture a weak general negative correlation among terms, the logistic normal can capture much richer dependency structure between terms according to the parameters of the Gaussian covariance matrix. The logistic normal class has a total of $\frac{1}{2}d(d+3)$ parameters compared to the $d+1$ of the Dirichlet.

Special case: Dirichlet approximation

A query model is often defined using a multinomial distribution over words. Since defining a query neighborhood amounts to specifying a prior over the query model, it is convenient to specify the query neighborhood in terms of the conjugate prior Dirichlet distribution.

We can closely approximate the Dirichlet distribution using the following formulas from Aitchison & Shen [Aitchison & Shen 1980] to convert a $d+1$ dimensional Dirichlet distribution with parameters $\boldsymbol{\alpha_i}$ to a logistic normal mean $\boldsymbol{\mu}$ and covariance matrix Σ (with

entries μ_i and σ_{ij}).

$$\mu_i = \Psi_0(\alpha_i) - \Psi_0(\alpha_{d+1}) \tag{4.21}$$

$$\sigma_{ii} = \Psi_1(\alpha_i) + \Psi_1(\alpha_{d+1}) \tag{4.22}$$

$$\sigma_{ij} = \Psi_1(\alpha_{d+1}) \qquad (i \neq j) \tag{4.23}$$

where $\Psi_0(z)$ and $\Psi_1(z)$ are the digamma and trigamma functions respectively[3].

When using a Dirichlet distribution for the query neighborhood, we use a Dirichlet prior β_U to control the sharpness of the overall distribution and thus the amount of variation in the sigma-points. When $\beta_U \gg 1$, the variants are only small adjustments to the original query. If $\beta_U \ll 1$, we get a mixture of variants, half of which are very close to a LOO sample and the other half to a TAT sample.

In this study, we use a simple variation strategy and allow $\boldsymbol{r}$ to vary according to a Dirichlet distribution $\boldsymbol{\alpha_r}$ having its mean at the center of the simplex. Thus, to use $\boldsymbol{\alpha_r}$ with the unscented transform we just approximate $\boldsymbol{\alpha_r}$ using Eq. 4.23, to obtain the logistic normal (μ_r, Σ_r).

Non-parametric query neighborhoods

The problem of estimating query neighborhood density is related to sensitivity estimation of text classifiers. As such, it is possible to construct non-parametric density estimates with some increase in computation cost.

In a recent example, Bennett [Bennett 2007] estimates the sensitivity of a classification function with respect to small changes in the target (query) point. He defines a local non-parametric density function for the query point based on its Voronoi cell, and computes the classification function for a small set of points sampled from the border of the Voronoi cell. The advantage of this approach is that it samples in directions that are locally dense, avoiding unlikely query points. A similar approach could be applied to retrieval, with the k query variants selected to maintain document ranking (say) close to that of to the original query.

In this chapter, we focus on simple parametric methods for generating query variants since these are computationally cheap to calculate and are sufficient to test the hypothesis that some form of query variation is useful. In future work it would be interesting to test

[3] $\Psi_0(z) = \Gamma'(z)/\Gamma(z)$ and $\Psi_1(z) = \Psi_0(z)'$

what, if any, improvements using non-parametric estimates might give.

4.2.5 Sigma-point sampling with the Unscented Transform

Assuming we have defined a parametric query neighborhood with the logistic normal, we derive a deterministic importance sampling strategy called *sigma-point sampling*, which is derived from a powerful method for computing statistics of non-linear functions of a random variable, called the *unscented transform* [Julier & Uhlmann 2002] (UT). Previously, the main use for the UT has been in particle filtering algorithms, where it has been very successful. The unscented transform both specifies a way to select query samples and also specifies the weights to use with the samples to calculate approximate lower moments of the output distribution, such as mean, variance, and skew.

The key idea of sigma-point sampling is to choose a small number of points that approximate the query neighborhood density. The sigma points are chosen such that their mean and variance are equal to the mean and variance of the underlying input distribution.

The basic unscented transform works as follows. Suppose $f(\theta)$ is a scoring function, and suppose we approximate the density $h(\theta)$ of θ as a d-dimensional Gaussian with mean μ and covariance matrix Σ. Then to approximate the expectation $\int f(\theta)h(\theta)$ we choose $2d$ points x_k for $k = 1, \ldots, 2d$ such that

$$x_k = \mu + (\sqrt{d\Sigma})_k \tag{4.24}$$

$$x_{d+k} = \mu - (\sqrt{d\Sigma})_k \tag{4.25}$$

where $(\sqrt{d\Sigma})_k$ is the k-th column of the matrix square root of Σ. The matrix square root is defined such that if UDU^T is the singular value decomposition of Σ, with $U = \{U_1, \ldots, U_d\}$ and $D = diag\{\lambda_1, \ldots, \lambda_d\}$ then $(\sqrt{d\Sigma})_k = \sqrt{\lambda_k}U_k$. The sample points x_k effectively summarize the mean and variance of $h(\theta)$ and are then used in the following Monte-Carlo-like approximation:

$$\int f(x)h(x)dx \approx \frac{1}{2d}\sum_{k=1}^{2d} w_k \cdot f(x_k). \tag{4.26}$$

This method can be generalized to include μ and scaled versions of x_k as additional sample points.

Given a set of $2d + 1$ sigma points $\{x_i\}$ in the input space, the unscented transform

defines the weight w_m of the sigma point at the mean to be

$$w_m = \frac{\kappa}{\alpha^2(d-1+\kappa)} + (1 - \frac{1}{\alpha^2}) \tag{4.27}$$

and the other $2d$ sigma points have weight

$$w_s = \frac{1}{\alpha^2(d-1+\kappa)} \tag{4.28}$$

where we set $\alpha = 1.0$, $\kappa = max(1, d-2)$. Using these weights, the unscented transform guarantees [Julier & Uhlmann 2002] that the mean and variance of $f(\theta)$ are given with accuracy up to second-order by

$$\mu = \mathbb{E}_h[f(x)] = \sum_{i=1}^{2d} w_{m:i} f(x_i) \tag{4.29}$$

$$\sigma^2 = \mathbb{E}_h[f^2(x)] - \mu^2 = \sum_{i=1}^{2d} w_{s:i} f(x_i) - \mu^2 \tag{4.30}$$

where the sigma points x_i and corresponding weights $w_{m:i}$ (for means) and $w_{s:i}$ (for variance) are derived based on the Gaussian input distribution $h(x)$. To review, the sigma-point sampling method gives us a provably good way of choosing samples in query space. We allow the query term weights to be distributed according to the logistic normal on the simplex, or as an approximate Dirichlet as a special case. Using the unscented transform, we can use these query samples to compute the expected value and variance of any non-linear scoring function of the query.

With this density defined over the input space, the unscented transform gives us a set of 'sigma points' in query space that can be used as query variants. These variants are provably good in the sense that, when used with the appropriate weights, we obtain approximations of the mean and variance of the scoring function that are accurate to second-order.

Sigma-point sampling is a theoretically well-founded strategy for sampling the query neighborhood. By 'well-founded' we mean that the estimates of important statistics of the scoring function such as the mean and variance are guaranteed to be accurate up to second-order. In other words, if the scoring function is quadratic, then the unscented transform is exact. An example of sigma-point query variants is shown in Table 4.1.

The processing cost of sigma-point sampling is not high, because for a query of N

Variant	Query terms
A1.	`foreign (0.333333); minorities (0.333333); germany (0.333333)`
A2.	`foreign (0.133447); minorities (0.133447); germany (0.733106)`
A3.	`foreign (0.458289); minorities (0.458289); germany (0.083422)`
A4.	`foreign (0.092386); minorities (0.660575); germany (0.247039)`
A5.	`foreign (0.660575); minorities (0.092386); germany (0.247039)`

Table 4.1: Example of five $(2N - 1)$ sigma-point query variants used for TREC topic 401, "foreign minorities germany". Relative term weights are shown in parenthesis next to each term. We could also make other choices, such as permutations of these weights.

terms, we need to perform a singular value decomposition of the $N - 1$ by $N - 1$ logistic normal covariance matrix; since N is typically 10 terms or less for typical queries, the covariance matrix is small.

One potential disadvantage is that we require more samples than the TAT and LOO methods: a total of $2N - 1$ samples instead of $N + 1$. For example, a five-word query would require 6 LOO samples (the original query plus 5 LOO variants), while sigma-point sampling would use 9 variants. Because most information retrieval queries are five terms or less, this difference will usually be small.

There are interesting directions for future work: sigma-point sampling is a general integration method not limited to use with queries. It could be used with document models to calculate expectations of common similarity or loss functions between queries and documents, with respect to mixtures of Gaussians or Dirichlets for example. We leave details for a separate study. For example, it would be interesting to evaluate how the number of sigma-point samples affects performance for different applications.

Figure 4.2 visualizes all three methods for a three-term query. The original query, with all term weights equal, is in the center of each simplex. The black dots denote query variants that result from each scheme. TAT sampling (top left) gives all the weight to one term per sample and sets the other term weights to zero. Leave-one-out (top right) sets one term weight to zero in from each sample and gives the rest equal weight. The sigma points method (second row) selects certain eigenvectors assuming the query was drawn from a logistic normal distribution. In effect this gives a 'softer' version of the other two discrete strategies combined. Also shown is the case where the initial query term weights are not all equal, to show how the sigma-point method adapts to the simplex geometry correctly.

function PreparePerturbationKernel(training_data)
$\hat{\theta} \leftarrow MLE\_Estimator(training\_data)$**;**
$V \leftarrow GeneratePerturbations(training\_data)$**;**
forall ***perturbations*** $v_k \in V$ **do**
  $\hat{\theta}_k \leftarrow MLE\_Estimator(v_k)$**;**
  forall ***elements*** x_i ***in finite domain*** $\mathcal{X}$ **do**
    $\phi_k(x_i) = (\sqrt{\hat{\theta}_k[x_i]} - \sqrt{\hat{\theta}[x_i]})/\sqrt{\hat{\theta}(x_i)}$ **/* Other feature mappings possible */**
  end
end
return ϕ

Function `PreparePerturbationKernel(`*training_data*`)`

$\phi \leftarrow$ PreparePerturbationKernel(training_data);
$r \leftarrow$ PerturbationKernel(ϕ,"watermelon", "banana");
function PerturbationKernel(ϕ, x_i, x_j**)**
$distance \leftarrow \sum_{k=1}^{|\phi|}(\phi_k(x_i) - \phi_k(x_j))^2$
return *distance***;**

Function `PerturbationKernel(`x_i, x_j`)`

A pseudocode version of the data perturbation kernel function and its setup function `PreparePerturbationKernel` are given in Figure PreparePerturbationKernel.

In the next sections we apply the methods of Section 4.2 to two important similarity estimation problems: term similarity, and language model similarity. The common theme is that we extend existing measures by replacing a single base statistic with an *expectation* over that base statistic with respect to the distribution over query functions.

4.3 Application: Term similarity

Statistical properties of words in text are a key ingredient in modern information retrieval methods. In this section we are concerned with estimating statistical properties of a very basic unit: pairs of terms. For example, we may wish to measure how a pair of terms is correlated, given a set of documents. From the earliest work on information retrieval, researchers have been interested in how pairs of words are related, and in automatic methods for quantifying term dependencies. Doyle [Doyle 1961] described a simple statistical dependency test which was essentially Pointwise Mutual Information (PMI). Doyle used PMI to find phrases and compound words (adjacent correlations), and term dependencies (proximal correlations), with the overall goal of creating term association maps or heirarchies to

assist with searching and browsing. Many other flavors of association measures, such like Expected Mutual Information (EMIM) [van Rijsbergen 1979] have also been explored: many of these are variations on the associations possible over the 2×2 contingency table of term occurrence. In addition to document frequency data, external resources such as WordNet [Resnik 1995] have been used to form or augment term similarity measures.

Much of this long history of term similarity has been applied to query expansion. In early work, Spärck Jones [Jones 1971] clustered words using word cooccurrence in documents, and used the clusters to expand queries. Term clustering has been used for both manual [Thompson & Croft 1989] and automatic [Qiu & Frei 1993] expansion. In the context of query expansion, there are two broad types of term similarity: *global* methods, which are query-independent and typically use statistics over the entire collection; and *local* methods, where the computation is specific to a given query, such as by using the top ranked documents of the query. Xu and Croft [Xu & Croft 2000] combined global and local methods in *local context analysis* [Xu & Croft 2000] for automatic query expansion. By improving query-specific term similarity methods, therefore, we can potentially improve important information retrieval methods like automatic query expansion that rely on them.

As we noted earlier, the use of query variants has certain advantages over co-occurrence statistics that treat the query terms separately. Because we only perturb part of the original query, the covariance data for a term with a given query term has the remaining query terms to use for context. This makes the perturbation kernel more conservative in finding good related words, accounting for polysemy by requiring that dependencies with multiple terms in context exist for the closest related words. This reliance on multiple relations with the query is reflected in the fact that the (Euclidean) distance in perturbation space that defines word similarity for an n-word query is the sum of squares of n perturbation features, each of which measures the interaction between the target word and a query term. Words will have a very low Euclidean distance only when the differences of many perturbation features are low.

4.3.1 Mathematical formulation

For term similarity, the task input domain $\mathcal{V}$ is a finite vocabulary set of terms. We suppose we have an unperturbed query $\boldsymbol{q}$ with language model $\theta_{\boldsymbol{q}}$. For each query variant $\boldsymbol{q}_i$ of $\boldsymbol{q}$, we use the corresponding top-retrieved documents $\mathcal{D}_i$ and their scores to estimate a generative Relevance Model [Lavrenko 2004] with parameters θ_i. We can now quantify the effect on

the model probability $p(w|\theta_i)$ of word w due to the perturbation (query variant) $\boldsymbol{q}_i$, which we denote by the mapping $\phi_i(w)$ for w.

There are many ways to define $\phi_i(w)$, but one plausible approach starts by calculating the *odds ratio* $o_i(w)$

$$o_i(w) = \frac{p(w|\theta_i)}{p(w|\theta_{\boldsymbol{q}})} \tag{4.31}$$

We can then consider the mapping

$$\begin{aligned}\phi_i(w) &= 2\log o_i(w) &(4.32)\\ &= 2\log \frac{p(w|\theta_i)}{p(w|\theta_{\boldsymbol{q}})} &(4.33)\\ &= 2(\log p(w|\theta_i) - \log p(w|\theta_{\boldsymbol{q}})) &(4.34)\end{aligned}$$

This gives the relative change in log-likelihood of term w in model θ_k (from query variant $\boldsymbol{q}_k$) compared to the Relevance Model $p(\cdot|\theta_{\boldsymbol{q}})$ for the original query. Essentially, this results in exactly the likelihood displacement measure of Cook (see Eq. B.2 in Sec. B), which shows a connection between the idea of a perturbation kernel and previous work in statistics on local influence, described further in Appendix B.

It is instructive to analyze how the canonical integral in Eq. 4.6 can be decomposed when substituting Eq. 4.32 into Eq. 4.6 we obtain the expansion

$$\begin{aligned}\Delta(w_1, w_2) &= \int_{\mathcal{F}} (\log \frac{p(w_1|\theta_k)}{p(w_1|\theta)} - \log \frac{p(w_2|\theta_k)}{p(w_2|\theta)})^2 dQ(f) &(4.35)\\ &= \int_{\mathcal{F}} \log^2 \frac{p(w_1|\theta_k)p(w_2|\theta)}{p(w_2|\theta_k)p(w_1|\theta)} dQ(f) &(4.36)\end{aligned}$$

We can define a factor

$$\Delta_0(w_1, w_2) = \log \frac{p(w_2|\theta)}{p(w_1|\theta)} \tag{4.37}$$

that is independent of query variants, and the complementary factor

$$\Delta_k(w_1, w_2) = \log \frac{p(w_1|\theta_k)}{p(w_2|\theta_k)} \tag{4.38}$$

that does not involve the original query. Using these, we can then rewrite $\Delta(w_1, w_2)$ as

$$\Delta(w_1, w_2) = \Delta_0(w_1, w_2)^2 + 2 \cdot \Delta_0(w_1, w_2) \int_{\mathcal{F}} \Delta_k(w_1, w_2) dQ(f) \tag{4.39}$$

$$+ \int_{\mathcal{F}} \Delta_k(w_1, w_2)^2 dQ(f) \tag{4.40}$$

Note that the function $\Delta(w_1, w_2)$ may not be a strict metric: for example, we can have $\Delta(w_1, w_2) = 0$ when w_1 and w_2 are different.

This analysis shows how adding query variants allows us to distinguish between two terms w_1 and w_2 having the same or very similar probabilities in the original relevance model θ: in that case, $\Delta_0(w_1, w_2) \approx 0$ and so $\Delta(w_1, w_2)$ becomes a function of the additional variant model alone.

Writing the term similarity measure as an integral (or sum of integrals) allows us to apply the sampling-based integration methods of Chapter 2. In particular, we can use the sigma-point sampling approximation to Eq. 4.39 with n query variants $\boldsymbol{q}_1 \ldots \boldsymbol{q}_n$ with sample weights w_k as defined in Eq. 4.27 and Eq. 4.28. Using the resulting Relevance Models $\theta_1 \ldots \theta_n$ to define the Δ_k, the term similarity function using the data perturbation kernel is given by:

$$\Delta(w_1, w_2) \approx \Delta_0(w_1, w_2)^2 + 2 \cdot \Delta_0(w_1, w_2) \sum_k w_k \Delta_k(w_1, w_2) \tag{4.41}$$

$$+ \sum_k w_k \Delta_k(w_1, w_2)^2 \tag{4.42}$$

Hellinger feature mapping As an alternative to Eq. 4.32 where we transformed $o(w)$ by $\log u$, we can use the transform of the function $\sqrt{u} - 1$, which has a similar shape, to obtain

$$\phi_k(w) = \sqrt{o_k(w)} - 1 \tag{4.43}$$

$$= \sqrt{\frac{p(w|\theta_k)}{p(w|\theta_{\boldsymbol{q}})}} - \sqrt{\frac{p(w|\theta_{\boldsymbol{q}})}{p(w|\theta_{\boldsymbol{q}})}} \tag{4.44}$$

$$= \frac{\sqrt{p(w|\theta_k)} - \sqrt{p(w|\theta_{\boldsymbol{q}})}}{\sqrt{p(w|\theta_{\boldsymbol{q}})}} \tag{4.45}$$

which is exactly the mapping used by the leave-one-out kernel (see Eq. 4.12) that is based on the Hellinger norm.

4.3.2 Visualizing perturbation similarity

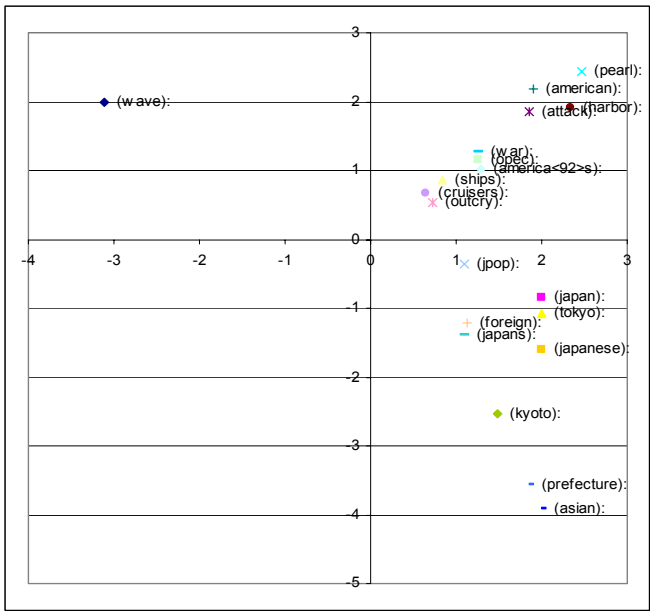
A simple example of a term similarity mapping using query variants for 20 individual feedback terms is shown in Figure 4.5. (The symbols used to plot the words have no special meaning and can be ignored.) The x-axis maps $\phi_1(w)$ and the y-axis maps $\phi_2(w)$ where

$$\phi_k(w) = \frac{\sqrt{p_{\alpha_k}(w)} - \sqrt{p_{\mathbf{q}}(w)}}{\sqrt{p_{\mathbf{q}}(w)}} \tag{4.46}$$

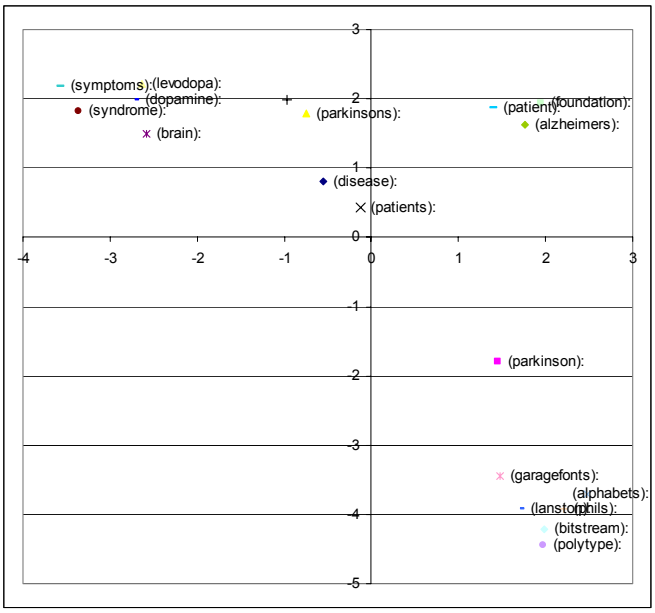
where the perturbation strategy is leave-one-out. In other words, ϕ_k indicates change in feedback model probability when one word, (e.g. "wave" in the "japanese wave" topic) is given greatly reduced weight in the query. The probability given the original unmodified query is the baseline mean level, represented by the origin. Thus, terms that have similar responses to the same query perturbations are close in this space. By ‘close’, we mean simple Euclidean distance between two points.

The perturbation feature mapping provides several types of useful information about the query and feedback model terms and their relationships. First, it gives a clustering of terms in the feedback model. For example, in Subfigure 4.5a the "pearl harbor" noise cluster has been separated from the other terms and placed in the NE quadrant, while the SE quadrant brings together terms related to "japanese", such as "asian" and "prefecture". The *absence* of related terms is also evident: no term is close to "wave" in aspect space. This may be evidence that this is a singularly important term to the query because it is difficult to find related words, and thus must be preserved and given significant weight.

Second, the global location of the clusters provides some indication of their potential relevance to the original query topic. The Euclidean distance from the origin indicates the variance of each term’s estimated probability of relevance compared to the original query. The farther reaches of the NE quadrant contain terms whose probability *increased* significantly when either of the first two query terms was removed, making these terms less likely to be important to the original information need. On the other hand, terms in the NW and SE quadrants had lower probability when one of the corresponding terms was removed, providing evidence of partial relevance. Terms in the SW quadrant are less common but are the most likely to be strong additions to a feedback model since since they covary with

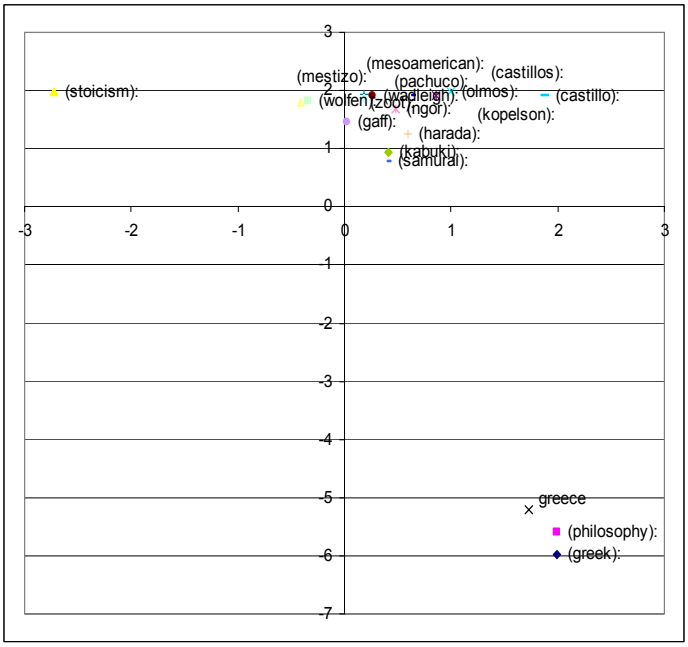


(a) Topic 491 japanese wave

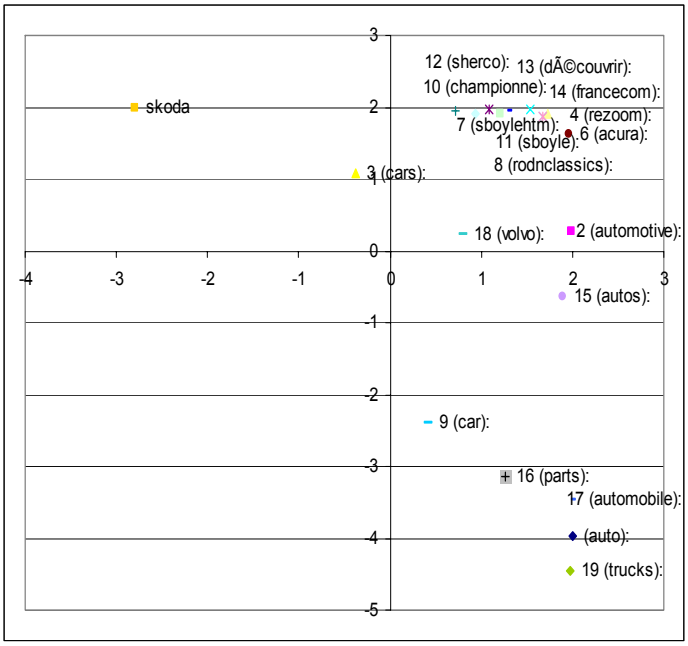


(b) Topic 406 parkinson's disease

Figure 4.5: The top 20 expansion terms for several TREC topics as they map to the first two co-ordinates in perturbation space. The *x*-axis represents the log of the relative change in probability for a word in the feedback model for the first query variant, compared to its probability in the feedback model for the initial query. Similarly, the *y*-axis shows the log change for the second query variants, and so on. Thus, terms whose probabilities respond similarly to the same query variants are close in this space.



(c) Topic 433 greek philosophy stoicism



(d) Topic 484 skoda auto

Figure 4.5: The top 20 expansion terms for several TREC topics as they map to the first two co-ordinates in perturbation space. Note how the mapping to perturbation space is effective in removing many expansion noise terms from the neighborhood of the original query terms, typically placing them in the upper right corner of this space. Close words in the upper right corner have been jittered apart for clarity.

both query terms and thus more likely to be related to both concepts.

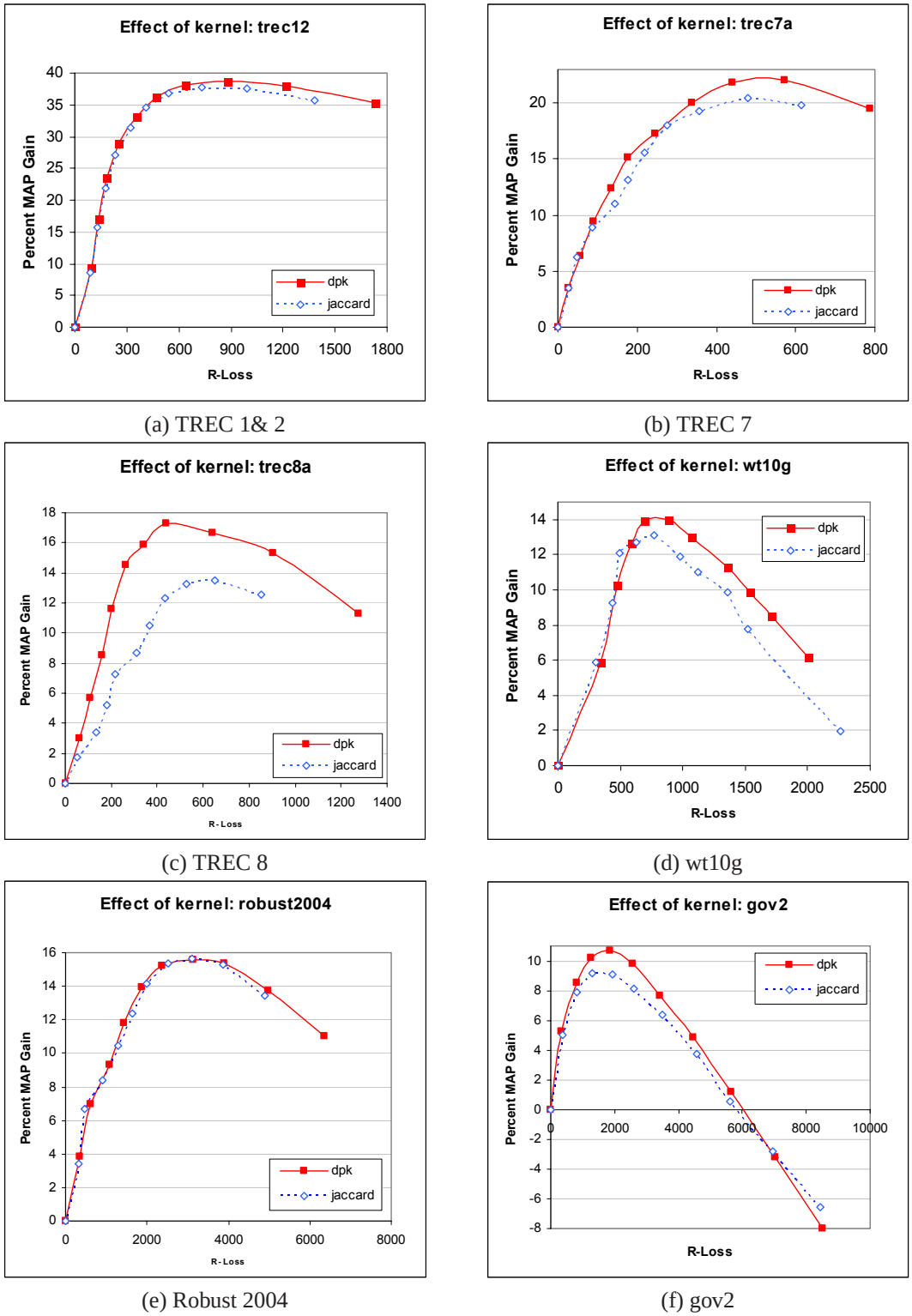
Third, the relative proximity of the original query terms to each other in perturbation space gives an indication for how phrase-like their behavior is. This reflects the fact that, if two or more terms form a single concept together, such that removing one of the terms results in substantially altering that concept – and thus the documents in which it is used – then the variation caused by these changes will be significantly disruptive no matter which term is removed and their statistics will be very similar.

4.3.3 Evaluation of kernel performance for query expansion

We now perform a task-based evaluation of the term clustering obtained with the perturbation kernel by applying it to query expansion. Later in this thesis, in Chapter 6, we will introduce a query expansion method called the *QMOD algorithm* that takes a kernel matrix as one of its inputs. Because it is very easy to switch in different kernels into the QMOD algorithm, we use that method here as our query expansion task. However, it is not important at this point to know any further details about the QMOD algorithm for the purposes of evaluating the relative performance of different kernels.

To visualize expansion performance we use the risk-reward tradeoff curve, described in Section 3.4.4. The curves for our standard six TREC topic sets are shown in Figure 4.6. For comparison, we chose a kernel derived from a term association measure that did not require the use of query variants and could be calculated from the initial set of top-retrieved documents. Recall that we are deriving term association statistics from a set of documents that is already biased toward the query terms, so that the number of documents *not* containing a query term is frequently zero, or close to zero. Thus, we need to use an effective term assocation measure that ignores this non-relevant negative information. The Jaccard measure is a simple, long-used term assocation measure that satisfies this property. Details on the Jaccard measure are given in Section A.2 of Appendix A. We also tried several other association measures, including expected and pointwise mutual information, but the Jaccard measure gave the best relative performance of these.

Figure 4.6 compares the risk-reward tradeoff curves using the perturbation kernel against the Jaccard distance, using the QMOD query expansion algorithm that will be introduced in Chapter 6. For four of the six collections (TREC 7, TREC 8, wt10g, and gov2) the perturbation kernel provides a small but consistent improvements over the Jaccard measure for virtually all values of α. At a setting of $\alpha = 0.5$, the improvements are largest for



(a) TREC 1& 2 (b) TREC 7

(c) TREC 8 (d) wt10g

(e) Robust 2004 (f) gov2

Figure 4.6: Risk-reward tradeoff curves for six TREC topic sets, showing the improved performance of the perturbation kernel compared to a Jaccard kernel on some collections. The solid (red) line is the curve given by the QMOD algorithm using the perturbation kernel. The dashed (pink) line uses the same QMOD algorithm and parameter settings, but substitutes a Jaccard kernel. Tradeoff curves that are *higher and to the left* give a better risk-reward tradeoff. Curves are plotted with points at α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.

TREC 8 and gov2. For TREC 8, the perturbation kernel gives a MAP gain of 14.5% with R-Loss of 262, while the Jaccard kernel gives a MAP gain of 8.68% with R-Loss of 307. For the gov2 collection, the perturbation kernel MAP gain is 9.78% with R-Loss of 2555, while the Jaccard kernel has MAP gain of 8.13% with R-Loss of 2605. For two of the collections (TREC 1&2 and Robust 2004), the performance of the two kernels is almost identical: TREC 1&2 shows only a tiny advantage for the perturbation kernel for $\alpha \geq 0.6$. The results suggest that the perturbation kernel gives the potential for useful gains on some collections, with little downside risk.

4.4 Application: Language model similarity and query difficulty

We now examine similarity functions over sets of *models* rather than sets of single parameters. Our goal will be to obtain a similarity function to compare a query model against a collection model as an estimate of query difficulty. We show that using query variants generalizes this clarity-type measure of query difficulty, by replacing a single similarity calculation with an *expected* distance from the query model to the collection model, with respect to the query neighborhood distribution.

4.4.1 Generalizing the query clarity score

The *query clarity* score was introduced by Cronen-Townsend and Croft [Cronen-Townsend & Croft 2002] as a way to estimate query difficulty by calculating the similarity of a given query's unigram model to the collection unigram model using KL-divergence. In this section we show how the clarity score generalizes to an expected similarity with respect to multiple query variants.

Let $\theta_{\boldsymbol{q}}$ represents the query model derived for the query $\boldsymbol{q}$, and let θ_i be the model derived for a query variant $\boldsymbol{q}_i$ taken in the query neighborhood Q. Suppose θ_C represents a language model describing the collection C. Then we define the *generalized clarity* score of a query $\boldsymbol{q}$ with respect to the collection C and query neighborhood measure Q as

$$A_{Q,C}(\boldsymbol{q}) = \mathbb{E}_Q[KL(\theta_{\boldsymbol{q}}||\theta_i)] + \mathbb{E}_Q[KL(\theta_C||\theta_i]) - 2\mathbb{E}_Q[\sqrt{KL(\theta_{\boldsymbol{q}}||\theta_i)KL(\theta_C||\theta_i)}] \qquad (4.47)$$

Thus, when the query is treated as a random variable, the distance from the query to the collection becomes an expectation over the query neighborhood Q.

If we choose the query neighborhood Q to be a point-distribution with all probability mass on the original query $\boldsymbol{q}$, so that

$$Q_{0,1}^{\boldsymbol{q}}(\theta) = \begin{cases} 1 & \text{if } \theta = \theta_{\boldsymbol{q}} \\ 0 & \text{otherwise} \end{cases} \tag{4.48}$$

then

$$\mathbb{E}_Q[KL(\theta_{\boldsymbol{q}}||\theta_i)] = KL(\theta_{\boldsymbol{q}}||\theta_{\boldsymbol{q}})$$

and generalized clarity reduces to the base clarity score for $\boldsymbol{q}$ and collection C:

$$A_C(\boldsymbol{q}) = KL(\theta_{\boldsymbol{q}}||\theta_{\boldsymbol{q}}) + KL(\theta_C||\theta_{\boldsymbol{q}}) - 2\sqrt{KL(\theta_{\boldsymbol{q}}||\theta_{\boldsymbol{q}})KL(\theta_C||\theta_{\boldsymbol{q}})} \tag{4.49}$$

$$= 0 + KL(\theta_C||\theta_{\boldsymbol{q}}) + \sqrt{0 \cdot KL(\theta_C||\theta_{\boldsymbol{q}})} \tag{4.50}$$

$$= KL(\theta_C||\theta_{\boldsymbol{q}}) \tag{4.51}$$

which is just the clarity score for query $\boldsymbol{q}$ and collection C. Thus, generalized clarity adds additional second-order information about $\boldsymbol{q}$ when the set of auxiliary queries of Q is not empty.

4.5 Evaluation of generalized clarity

In this section we evaluate the usefulness of using variant-based similarity measures on a language model similarity task, to estimate a clarity score.

4.5.1 General method

We evaluated performance on the following sets of TREC topics and collections: 51-150 (TREC-1&2), 201-250 (TREC-4), 351-400 (TREC-7), 401-450 (TREC-8), and 451-550 (wt10g, TREC-9&10). We also included the set of topics from the TREC Robust 2004 (301-450; 601-700) track. We chose these collections for their varied content and document properties. For example, wt10g documents are Web pages with a wide variety of subjects and styles while TREC-1&2 documents are more homogeneous news articles.

Indexing and retrieval was performed using the Indri system in the Lemur toolkit [Metzler & Croft 2004] [Lemur 2002]. Our queries were derived from the words in the title field of the TREC topics. Phrases were not used. We performed Krovetz stemming for all experiments with a stoplist of 419 common English words. Other details on the baseline

Collection	TAT	LOO	Sigma-point
TREC 1&2	0.311	0.338	**0.406**
TREC 4	-0.071	0.401	**0.465**
TREC 7	0.265	0.221	**0.268**
TREC 8	**0.446**	0.414	0.418
Robust 2004	0.178	0.212	**0.249**

Table 4.2: The effect of different query sampling strategies on the Kendall-τ correlation of generalized clarity with average precision.

feedback method and query syntax can be found in [Collins-Thompson & Callan 2007].

There is some inconsistency in the way query performance prediction is evaluated across studies. We choose one of the more consistent measures found in the various papers: Kendall's τ, which measures similarity between two rankings. In our study, this means we compared the ranking of topics sorted by a performance baseline such as average precision, with ranking of the same topics sorted by the query scoring function.

4.5.2 Effect of query sampling strategies

We evaluated how each of the three query sampling strategies – LOO, TAT, and sigma-point sampling – affected the quality of the generalized clarity estimate, which was measured by Kendall-τ correlation with average precision. The results are summarized in Table 4.2.

Sigma-point (SP) sampling achieved the highest score on 4 out of 5 collections, compared with LOO sampling (0 collections) and TAT sampling (1 collection). SP sampling achieved significant gains of +0.064 in Kendall-τ correlation on both TREC 1&2 and TREC 4, compared to LOO sampling. Sigma-point sampling (SP) was better than TAT and LOO, except for a small difference on TREC-8.

In general, SP sampling appears to have a useful gain in quality for this task, with a more consistent upside and smaller downside risk. However, the amount of gain is relatively small. This may be due to query length effects, or the need for additional parameter tuning for sigma-point query neighborhoods.

4.5.3 Effectiveness of generalized clarity score

Our goal in this section is not to claim the best general query difficulty method. Instead, we tested how useful the generalized clarity score (GC) was, relative to the baseline clarity

Collection	GC	C	C+GC
TREC 1&2: 51–150	0.406	0.339	**0.451**
TREC 4: 201–250	0.465	0.552	**0.555**
TREC 7: 351–400	0.268	**0.416**	0.383
TREC 8: 401–450	0.418	0.394	**0.448**
Robust 2004	0.249	0.361	**0.362**
wt10g: 451–550	0.238	0.275	**0.278**

Table 4.3: Kendall-τ rank-correlation with average precision of generalized clarity(GC), clarity score(C) and combined (C+GC) with average precision. Sigma-point sampling was used for query variation.

(C) method, and when combined with baseline clarity using simple interpolation (C+GC) with C and GC given equal weight.

A comparison of the Kendall-τ correlation for different methods and test collections is shown in Table 4.3. Although gains from adding GC as a feature were negligible for some collections, GC had a rank-correlation with average precision greater than 0.40 for the TREC 1&2, TREC 4, and TREC 8 collections. This led to a significant gain for the combined C+GC method for TREC 1&2 of 0.451, compared to baseline clarity (0.406), and for the TREC 8 collection (0.448 vs 0.394).

Conversely, adding GC almost never hurt baseline clarity performance. The one exception was the TREC 7 collection: baseline clarity was slightly higher (0.416) than the combined method (0.383). This is not unexpected since for TREC 7 generalized clarity had the lowest rank-correlation (0.268) with AP of all test collections.

The correlations obtained by the combined C+GC method compare favorably to state-of-the-art results. For example, on the collections shared with the evaluation (using 2 pooled systems) of [Aslam & Pavlu 2007] (which we call JS), C+GC score for TREC 7 was 0.383, compared to 0.436 (JS); TREC 8 C+GC was 0.448 compared to 0.443 (JS); Robust 2004 C+GC was 0.362 vs 0.393 (JS).

4.6 Related Work

We first review related work on distance measures and kernels, followed by recent methods for query-specific term similarity and estimating query difficulty.

4.6.1 Distance measures as probabilities

The distance between two data points or instances can be defined in terms of probability. In a classification setting, each instance is identified with a class label c, and so the probability might be the misclassification probability: the likelihood that two points actually have different class labels, rather than being in the same class [Minka 2000a]. Thus, the optimal choice for the 1-nearest neighbor of x is the neighbor x_L with lowest probability of misclassification. More generally, a canonical similarity function can be defined as the cost of mislabeling if two points are in different classes, with many different cost functions possible. Yianilos [Yianilos 1995] estimates the task distribution and views the Canonical Distortion Measure as an evidence ratio.

The idea of defining a distance by comparing probability distributions, also known as distributional similarity is discussed by Lee [Lee 1999]. Dillon *et al.* [Dillon et al. 2007] discuss word distance as probability of Type II error, where the distributions in question are the language models of the word contexts. Blanzieri and Ricci introduce a minimum risk distance measure [Blanzieri & Ricci 1999] that optimizes finite misclassification risk, extending an earlier nearest neighbor method of Short and Fukunaga [Short & Fukanaga 1980] that learned a reduced set of prototypes along with a local metric.

4.6.2 Kernels over probability densities

The probability product kernel and leave-one-out kernel described in Section 4.2.2 that integrate over input space are related to a broader family of kernels that involve inner products over probability densities.

Conditional symmetric kernels. Let h be a hidden variable whose values are drawn from a finite set H (this can be extended to continuous domains). Watkins [Watkins 2000] introduced *conditional symmetric kernels*

$$k(x, x') = \sum_{h \in H} p(x|h)p(x'|h)p(h) \tag{4.52}$$

The conditional symmetric kernel can be seen as a special case of data perturbation kernels if we identify the latent space H with the set of hypotheses of the 'true' information need expressed by some query variant. This requires the generative process $p(x|h)$ be known.

Marginalized kernels. If $p(h|x)$ is known, we can use *marginalized kernels* [Tsuda et al. 2002]. Suppose we have a kernel $k_Z(z, z')$ defined between the joint variables $z = (x, h)$ and $z' = (x', h')$. The marginalized kernel $k(x, x')$ is the expectation of k_Z over the hidden variable as follows:

$$k(x, x') = \sum_{h \in H} \sum_{h' \in H} p(h|x)p(h'|x')k_Z(z, z') \tag{4.53}$$

Zhao *et al.* [Zhao et al. 2006] add a time-dependent aspect to query-specific similarity using the marginalized kernel approach to model the evolution of click-through data.

Fisher kernels. The Fisher kernel, introduced by Jaakkola, Diekhans and Haussler [Jaakkola et al. 1999] is a special case of marginalized kernel. Our perturbation similarity measure is related to Fisher-type kernels that compare the sufficient statistics of generative models, in the following way.

We use term similarity as an example for an informal argument. Let $\hat{\theta}$ be the relevance model estimated using the original query. Each query variant $\boldsymbol{q}_i$ results in a corresponding relevance model θ_i. The *Fisher score* of a term x is a vector of derivatives: one for each latent variable.

$$U_x = g(\hat{\theta}, x) = \frac{\partial \log p_{\hat{\theta}}(x)}{\partial \theta} \tag{4.54}$$

and the Fisher kernel $\kappa(x_1, x_2)$ between x and y is given by

$$\kappa(x, y) = U_x \boldsymbol{I}^{-1} U_y. \tag{4.55}$$

Let $\theta_k = \hat{\theta} + \epsilon_k$, and suppose that $||\epsilon_k|| = \delta_k$ for some norm. The task function $f_k(w)$ we use for term similarity can be rewritten as

$$f_k(x) = \log \frac{p(x|\theta_k)}{p(x|\hat{\theta})} \tag{4.56}$$

$$= \delta_k \cdot \frac{\log p(x|\hat{\theta} + \epsilon_k) - \log p(x|\hat{\theta})}{\delta_k} \tag{4.57}$$

$$\approx \delta_k \cdot \frac{\partial \log p_{\hat{\theta}}(x)}{\partial \theta_k} \tag{4.58}$$

Assuming the magnitude of the δ_k are roughly δ for all k variants, we can denote $f(x)$ as the vector of $f_k(x)$ for all k, and it is evident that $f(x) \propto U_x$, i.e. that the vector of resulting perturbation scores is a type of Fisher score. However, unlike the classic Fisher kernel,

calculating the perturbation kernel does not assume the underlying likelihood function is differentiable, making it applicable to a wider family of generative models.

4.6.3 Query-specific term similarity and clustering

The basic concept of query-specific similarity measures has existed for decades. Preece [Preece 1973] and later Willett [Willett 1985] proposed improving document classification and clustering by including the query as context information. Tombros and van Rijsbergen [Tombros & van Rijsbergen 2001] proposed and tested the effectiveness of query-specific (also called query-sensitive) measures. Such measures are biased in favor of pairs of documents that jointly contain attributes (e.g. terms) of the query. According to Tombros and van Rijsbergen's hypothesis, "pairs of relevant documents will exhibit an inherent similarity which is dictated by the query itself (that) conventional measures... such as the cosine coefficient, can not detect..." They measured the effectiveness of a similarity measure by examining the degree to which relevant documents were brought closer together (compared to non-relevant ones). They found that simple query-sensitive measures gave significant improvement over traditional cosine similarity on several small test collections.

While the idea of query-specific similarity measures is not new, research on such algorithms has continued to be of great interest in areas such as Web search, which needs precise ways to compare not only documents but summaries, online ads, alternate queries, and other short snippets of text. Recent extensions for Web documents look beyond the basic query terms to other features related to the query such as clickthrough data, related entries in query logs, and so on.

Recently a few other kernels have used expansion methods to obtain more context for a similarity comparison. The kernel of [Sahami & Heilman 2006] compared two text snippets by computing the inner product of the query expansions that result by considering each text snippet as a Web query. Each query expansion is represented as a normalized centroid of the *tf.idf* vectors for the corresponding top-ranked documents. [Metzler et al. 2007] subsequently evaluated a set of related measures to perform query-query comparisons. In comparison to these, the perturbation kernel has focused on representing and comparing individual words, not snippets, according to each word's of perturbation features. There is, however, a natural extension for using perturbation kernels to compare text snippets, by representing each snippet as the union of the perturbation features derived for the snippet's individual words. In a way, this can be seen as a generalization of existing expansion-based

methods that adds sensitivity information to the basic expanded representation. We leave this as a subject for future work.

4.6.4 Other statistical text similarity methods

Recent work has begun to examine the implications of treating queries more fully as random variables. For example, Dillon *et al.* [Dillon et al. 2007] proposed a method for taking expected distances on word histograms with respect to a word translation model. They applied their method to text classification with generally positive improvements in classification accuracy. In a more general framework, Yih and Meek [Yih & Meek 2007] combine multiple types of similarity estimates, using logistic regression to learn combinations of similarity measures for query suggestion.

Related to the Fisher-type kernels of Section 4.6.2, [Lafferty & Lebanon 2002] introduced *information diffusion kernels* over statistical parameter spaces. In the case of discrete data such as text, information diffusion kernels can be used with the multinomial distribution to compute the geodesic distance between multinomial parameters θ_1 and θ_2 estimated from two text passages (e.g. using maximum likelihood). It would be interesting to explore the connections between this family of kernels and the generalization of perturbation kernels to text snippets described above.

Ando et al. [Ando et al. 2006] used leave-one-out (LOO) variants for performing query expansion in the TREC 2005 Genomics track. Most significantly, they reiterate the connection between multi-task learning and using auxiliary queries that was previously stated in general form by Baxter [Baxter 1997]. However, their work is not focused on metric learning for general IR objects, except in the sense that an enhanced query model provides a better similarity function for retrieval. Instead, they focus on estimating an improved query model by calculating a Rocchio-type feedback model using the length-normalized average of the positive examples.

4.6.5 Query difficulty

Recent studies point toward the conclusion that estimating *variance* is an important facet of predicting query performance. More specifically, the sensitivity of some aspect of the retrieval process to variation in input or model parameters has been shown to be effective in varying degrees. This includes variance of results ranking (by varying document models) [Zhou & Croft 2006], query variation [YomTov et al. 2005], query term *idf*

weights [Vinay et al. 2005] and document scores [Diaz 2007b].

Our method is in the query variation group. Previously, we showed that combining the results from multiple query variants improved the robustness and precision of a strong baseline pseudo-relevance feedback (PRF) method. Our approach has several differences from that of [YomTov et al. 2005]. First, we do not require a training phase. Second, our best query variation strategy is based on sigma-point sampling, not term-at-a-time. Collins-Thompson & Callan applied query variation to enhance the robustness of pseudo-relevance feedback [Collins-Thompson & Callan 2007].

Aslam & Pavlu [Aslam & Pavlu 2007] introduce variation by combining TREC runs from multiple systems for the same query. Their difficulty prediction statistic, which achieves impressive results with multiple systems, has close connections with our generalized clarity statistic: both methods measure the distance between the language models of the top-retrieved document sets obtained from either query variants (our system) or retrieval function variants (Aslam & Pavlu). Our interest is in the performance achievable with a single system, so it would be very interesting to understand more about how the variation achievable with multiple systems could be at least partially achieved with the correct types of perturbation to a single system.

4.6.6 Other uses of data perturbation

In the field of recommender systems (collaborative filtering, or CF) methods for quantifying the *influence* of a user were introduced by Rashid *et al.* [Al Mamunur Rashid & Riedl 2005][4]. In particular, for a given user U they measure the net change in ratings predictions for all other users, caused by leaving out the observed data for U. Change is measured by either total change across all items (NPD), or bucketed into unique users (a measure they term NUPD). This is attractive because it makes few assumptions about the underlying CF algorithm, although it can be computationally expensive. Mathematically, the influence $\mathcal{I}(u)$ of user u in user-set U is a sum over all items a (from item-set A)

$$\mathcal{I}_\delta(u) = \sum_{v \in U, v \neq u} \sum_{a \in A} [|\hat{p}_v^{(u)}(a) - \hat{p}_v(a)| \geq \delta] \tag{4.59}$$

where δ is a threshold that specifies the change needed for the smallest possible rating adjustment. Such influence methods are actually an instance of a general class of perturbation

[4]Thanks to Sean McNee for this connection.

techniques developed earlier in the statistics literature, which are described in Appendix B.

Other interesting uses of perturbations can be found in graph theory. For example, [Prabhu & Deo 1984] examine the extent to which two graphs can be discovered to be isomorphic by perturbing each graph G by adding a vertex and connecting it to all other vertices of G. The characteristic polynomials of the perturbed graphs $G_1 + v$ and $G_2 + v$ are then compared. There seem to be deeper connections here to kernel methods which have yet to be fully explored.

The idea of performing tests to measure the sensitivity of a model to perturbations in the training data or model parameters has been of interest to statisticians for some time, and is discussed further in Appendix B.

4.7 Conclusions

The data perturbation kernel is a useful tool for comparing the similarity of elements in a domain $\mathcal{X}$ when we have a probability density over the entire space $\mathcal{X}$ that is derived from using a possibly very small subset of it as training data. Similarity between elements is induced with respect to small perturbations in the training data. While the probability product space family of kernels assign exactly one probability density to each point in the input space $\mathcal{X}$ and then integrate over $\mathcal{X}$, our approach is essentially the dual: Each input point is identified with multiple probability densities evaluated at that point, and we integrate over probability density space. By treating the query as training data chosen from a discrete vocabulary space, we can obtain a query-specific similarity measure between words or language models by running a small number of query variants. Since we already use the same query variants to obtain multiple feedback models, there is essentially no additional cost to using this similarity measure for retrieval within our framework.

Using the data perturbation kernel on the space of language models generalizes the existing query clarity measure by adding a sensitivity component. This suggests the gain in Kendall tau with average precision may be a useful test for the IR *cluster hypothesis* in a given collection and set of queries. Furthermore, when this situation exists the method we use to explore the query neighborhood becomes more important, since our results will be more sensitive to the results of that method. Indeed, on those collections we observed some improvement moving from a term-at-a-time strategy to sigma-point sampling which used more query points to explore the neighborhood.

The data perturbation kernel is very simple to apply but has surprisingly deep connec-

tions to multi-task learning and Fisher-type kernels. We suspect there are also fundamental connections to spectral clustering methods but leave this analysis for future work. In the next chapter, the term similarity measure based on the perturbation kernel is used as an important ingredient in selecting effective query models in an optimization framework.

Chapter 5

Convex Optimization

The second part of this thesis, on optimization algorithms, forms the capstone of our work on estimating statistical models for robust retrieval. Here we apply the sampling-based estimators introduced in Chapter 3 and the data perturbation kernels described in Chapter 4 to form objectives and constraints within a general risk framework for estimating retrieval models. This novel optimization approach allows us to estimate good query models under a variety of useful retrieval scenarios.

One strength of this approach is that we can easily control the trade off between competing objectives and constraints such as maximizing expected utility of the model versus the risk of multiple sources of evidence that the model is based on. We can also model the dependencies between the sources of evidence themselves, or between the optimization objectives and multiple arbitrary constraints such as a minimum or maximum number of terms in the model, or the consistency of aspect coverage of the information need. As we show in Chapter 6, when evidence uncertainty and its relation to optimizing the objective is ignored, the result will be less robust query models.

We can break the problem of finding 'good' query models into two parts. First, we must specify the *objective functions* that describe the properties of the model we want to optimize. For example, we may wish to maximize the expected relevance score of the query model. Second, there may be any number of *constraints* that the model must also satisfy. For example, there might be a performance cost for adding an expansion term to a query, causing us to prefer queries with fewer terms. Another constraint might be that we prefer a query model that is more robust, i.e. less likely to hurt unexpanded performance, in exchange for lower average precision. Ideally, we not only want to find optimal query

models with respect to the objectives and constraints, but also have simple ways for users to specify how they want to trade off between competing objectives.

We give a detailed development of convex programming methods to address these problems in Chapter 6. In this chapter, we give some preliminary motivations and background for using convex optimization methods for information retrieval problems.

5.1 Optimization methods

Given the many possible objectives and constraints involved in choosing a query and the trade-offs possible among these factors, the problem of finding a 'good' query model is complex. However, there is a principled framework that allows us to structure the problem by making our assumptions clear and allowing us to control how we manage competing tradeoffs between objectives, while providing efficient computational methods to find solutions. This is the approach known as *convex optimization* (CO).

We will show that even simple CO methods, with easy-to-understand objectives and constraints, help us balance the various trade-offs required of the optimal query model, such as the trade-off between expected return (a good feedback model) and model variance (the amount of harm if wrong). Typically, our optimization will embody a basic tradeoff between wanting to use evidence that has strong expected relevance (such as highly-ranked documents, or highly-weighted expansion terms), and the *variance* or *risk* of using that evidence, or variance in covering the query aspects.

We now give some background on optimization and introducing basic terminology and concepts that we will use to specify objectives and constraints. An excellent resource on convex optimization theory and practice is Boyd & Vandenberghe [2004] and we follow their notation below.

5.1.1 General optimization problems

The general optimization problem is to find an x that minimizes the value of the function $f_0(x)$ for all eligible x that must satisfy the conditions $f_i(x) \leq 0$ and $h_i(x) = 0$. We use the

following notation:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{5.1}$$

to describe the problem, with $x \in \mathbb{R}^n$ being the *optimization variable* and the function $f_0(x) : \mathbb{R}^n \to \mathbb{R}$ being the *objective function*. The objective function is also sometimes called the *cost function*. The conditions $f_i(x) \leq 0$ and $h_i(x) = 0$ are called the *inequality constraints* and *equality constraints* respectively, with $f_i(x) : \mathbb{R}^n \to \mathbb{R}$ and $h_i(x) : \mathbb{R}^n \to \mathbb{R}$ being the corresponding *constraint functions*.

We denote the *domain* of the optimization problem as $\mathcal{D}$, which is the set of all x for which the objective function and all constraint functions are defined. A point $x \in \mathcal{D}$ is called *feasible* if all constraints $f_i(x) \leq 0$ and $h_i(x) = 0$ are satisfied at x. The optimization problem 5.1 itself is said to be *feasible* if at least one feasible point x exists. Otherwise, it is called *infeasible*, in which case we adapt the convention that the optimal value $x^\star$ of $x = \infty$. For any feasible points x_k with $f_0(x_k) \to \infty$ as $k \to \infty$ then the optimal value $x^\star = -\infty$, and we call the problem *unbounded below*. An optimization problem is *solvable* if there exists a feasible point for which the minimum of the objective function is attained.

An important special case of the general optimization problem 5.1 is the *feasibility problem*, in which the objective function $f_0(x) = 0$, i.e. is identically zero for all inputs. In this case, the optimal value of $f_0(x)$ is either zero if a feasible x exists, or ∞ if the feasible set is empty. This problem is written as

$$\begin{aligned} \text{find} \quad & x \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{5.2}$$

The feasibility problem is a consistency test of the constraints; if consistency is satisfied, we obtain a feasible point.

Although we have presented optimization in terms of minimization, naturally we can also form a corresponding maximization problem simply by minimizing $-f_0(x)$ with respect to the same constraints. In these cases, the objective function $-f_0(x)$ is also known as

the *utility* function.

5.1.2 Convex optimization

We first give a brief review of convex sets and functions. Informally, a set C is *convex* if every point in C can be "seen" from any other point in C. That is, there is an unobstructed straight path between them that lies entirely within C. Mathematically, for any $x, y \in C$ and $\alpha \in [0, 1]$, then

$$\alpha x + (1 - \alpha)y \in C. \tag{5.3}$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if its input space (domain) $\mathcal{D}$ is a convex set, and for any points $x, y \in \mathcal{D}$ the following holds for $\alpha \in [0, 1]$:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y). \tag{5.4}$$

If f is differentiable, then a very important property of convex functions is as follows: f is convex if and only if its domain $\mathcal{D}$ is convex and

$$f(y) \geq f(x) + \nabla f(x)^\mathsf{T}(y - x) \tag{5.5}$$

for all $x, y \in \mathcal{D}$. This implies that when $\nabla f(x) = 0$, then $f(y) \geq f(x)$ for all $y \in \mathcal{D}$ and x gives a global minimum for f.

A *convex optimization problem* is an optimization problem in which the following requirements hold.

- The objective function $f_0(x)$ is convex.
- The inequality constraint functions $f_i(x)$ are convex.
- The equality constraint functions are affine, i.e. have the form $h_i(x) = a_i^\mathsf{T} x - b_i$.
- The variable $x \in S$ where S is a convex set.

We write the standard form of a convex optimization problem as:

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq 0, \quad i = 1, \ldots, m \\ &&& a_i^\mathsf{T} x = b_i, \quad i = 1, \ldots, p \end{aligned} \tag{5.6}$$

R

Convex optimization has attractive theoretical properties. For example, a key property of convex problems is that, if we have a locally optimal solution point, then it must also be globally optimal. This means we will not get 'stuck' at a local maximum point and will not have to search exhaustively for a global solution. Also, convex optimization is tractable: solving can be done with polynomial complexity in the number of constraints and variables.[1]

In addition, certain meta-problems become easier when formulated as a convex program. Finding *robust* solutions to convex problems can often be also easily expressed as a CP. This is discussed further in Section 5.2.4. Also, many hard problems can be 'relaxed' to the form of a convex problem with known approximation bounds. The graph labeling problem discussed in Section 6.1 that we use for query model estimation is one application of convex relaxation.

5.1.3 Convexity of common retrieval functions

We note that many functions used as important scoring objectives or constraints in information retrieval are convex. For example, we state without proof the fact that for a given query model $\theta_{\boldsymbol{q}}$, the KL-divergence $KL(\theta_{\boldsymbol{q}}||\theta_{\boldsymbol{d}})$, an important comparison function in the language modeling approach to retrieval, is a convex function of $\theta_{\boldsymbol{d}}$. Similarly, the dot product of a given query vector with a document vector is always convex.

In fact, many common ranking functions are convex functions even in more complex query formulations. For example, the Indri [Metzler & Croft 2004] retrieval system makes use of an inference network constructed from term evidence nodes and operator nodes. A common query operation is the `#combine` operator

$$f_{\texttt{\#combine}}(b) = \prod_{i=1}^{n} b_i^{1/n} \tag{5.7}$$

which calculates the geometric mean of the input beliefs, and the variant the `#weight` operator

$$f_{\texttt{\#weight}}(b, w) = \prod_{i=1}^{n} b_i^{\frac{w_i}{\sum_k w_k}} \tag{5.8}$$

[1] Importantly – since many useful convex programs are non-smooth in practice – this can often be proven for many problems whether or not the objective and constraints are differentiable.

and the `#max` operator

$$f_{\texttt{\#max}}(b) = \max_{i=1\ldots n} b_i. \tag{5.9}$$

These are all convex functions of the belief inputs b_i. When certain combinations of operators are used to form a more complex query, the resulting ranking function is also convex. This is because nesting of convex functions (mathematically, called function composition) can result in convex functions under fairly broad conditions. As an example, the structured query (such as we might use to perform query expansion) of the form

```
#weight( 0.7 #combine(estonia economy) 0.3 #weight( 0.3 estonia 0.2
        economy 0.1 kroons 0.1 tallinn 0.1 baltic))
```

is a convex function of the concept weights, because the combined function is the composition of

$$f = f_{\texttt{\#weight}}(\{f_{\texttt{\#combine}}(b), f_{\texttt{\#weight}}(b)\}, w) \tag{5.10}$$

Since the function $f_{\texttt{\#weight}}$ is a non-decreasing function of its input, the resulting composed function is still convex.

The fact that many common retrieval scoring or comparison functions are convex means it is possible to use convex programming techniques to optimize over retrieval objects or actions in a realistic way.

5.2 Convex program families

The following are some basic forms of convex programming that we summarize as background information.

5.2.1 Linear programming

A linear program (LP) has objective and constraints that are all affine. The standard form for a LP is

$$\begin{array}{ll} \text{minimize} & c^{\mathsf{T}}x + d \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array} \tag{5.11}$$

where $c, d \in \mathbb{R}^n$,G $\in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{p\times n}$. The feasible set of an LP is some polyhedron $\mathcal{P}$ as defined by the constraints, and we are essentially finding the optimal point $x^\star \in \mathcal{P}$ that is farthest in the direction of the vector $-c$.

5.2.2 Quadratic programming

An important subclass of convex program, called a *quadratic program* (QP) is more general than linear programs because it allows a (convex) quadratic objective function with affine constraints. The standard form for a QP is

$$\begin{array}{ll} \text{minimize} & 1/2x^\mathsf{T}Px + q^\mathsf{T}x + r \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array} \tag{5.12}$$

where $P \in \boldsymbol{S}^n_+$, $G \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{p\times n}$. The notation $P \in \boldsymbol{S}^n_+$ denotes that the matrix P is in the family of positive semi-definite matrices.

If we allow (convex) quadratic inequality constraints instead of affine, we obtain the more general class of *quadratically constrained quadratic program* (QCQP).

$$\begin{array}{ll} \text{minimize} & 1/2x^\mathsf{T}P_0x + {q_0}^\mathsf{T}x + r_0 \\ \text{subject to} & 1/2x^\mathsf{T}P_ix + {q_i}^\mathsf{T}x + r_i \le 0, \quad i = 1,\ldots,m \\ & Ax = b \end{array} \tag{5.13}$$

where $P_i \in \boldsymbol{S}^n_+, i = 1,\ldots,m$. A QCQP minimizes a quadratic objective over a feasible set that is the intersection of ellipsoids. If we have $P_i = 0, i = 1,\ldots m$ this reduces to a simple QP, while taking $P = 0$ gives a linear program as a special case.

5.2.3 Second-order cone programming

A more general class of convex program called a *second-order cone program* (SOCP). For an SOCP we allow constraints of the form

$$||Ax + b|| \le c^\mathsf{T}x + d \tag{5.14}$$

with $A \in \mathbb{R}^{k \times n}$. Note the right-hand side allows an affine function instead of the constant d found in QCQPs. Thus a general second-order cone program (SOCP) has the form

$$\begin{array}{ll} \text{minimize} & c^{\mathsf{T}} x \\ \text{subject to} & \|A_i x + b_i\| \leq c_i{}^{\mathsf{T}} x + d_i, \quad i = 1, \ldots, L \\ & Fx = g \end{array} \tag{5.15}$$

Raw

where $\|\cdot\|$ is the Euclidean norm. When $c_i = 0, \quad i = 1 \ldots m$ the SOCP reduces to a QCQP as a special case. If $A_i = 0, \quad i = 1 \ldots m$, then we obtain a general linear program (LP).

The norm constraint in Eq. 5.14 is not limited to the Euclidean norm: it can be naturally generalized [Alizadeh & Goldfarb 2001] to a p-norm

$$\|x\|_p = \|x\|_{l/m} = (\sum_{i=1}^{n} |x_i|^{l/m})^{m/l} \tag{5.16}$$

where $p = l/m$ for positive integers l, m. The p-norm arises, for example, in the *extended Boolean model* of retrieval. [Salton et al. 1983]. The Euclidean norm corresponds to the case when $l = 2, m = 1$. Essentially, the value of p controls the 'softness' of the matching function, with $p \to \infty$ approaching the strict Boolean model, and $p \to 1/2$ approaching standard vector space similarity. The SOCP constraint

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^{l/m})^{m/l} \leq t \tag{5.17}$$

can be written as a set of second-order cone and linear inequalities, namely

$$\begin{aligned} -t^{\frac{l-m}{l}} s_i^{\frac{m}{l}} &\leq -x_i \\ t^{\frac{l-m}{l}} s_i^{\frac{m}{l}} &\leq x_i \\ s_i &\geq 0 \\ \sum_{i=1}^{n} s_i &\leq t \\ t &\geq 0 \end{aligned} \tag{5.18}$$

Such extensions increase the space of possibly useful SOCP programs for information re-

trieval situations.

5.2.4 Robust optimization

In *robust optimization*, we optimize for the worst-case scenario for an underlying simpler problem by expressing bounded uncertainty in the objective and constraint functions. This is also known as a *minimax* policy, because we are minimizing the maximum loss possible over the data, where the minimization is taken over all feasible sets of constraints. It is not surprising that, to formulate query models with good worst-case performance, we will examine robust optimization methods.

We perform robust optimization by making the objective and constraint functions $f_i(x)$ functions of not only the optimization variable $x \in \mathbb{R}^n$, but also a parameter vector $u \in \mathbb{R}^k$ that is a random variable that captures this uncertainty according to a specified probability distribution. The standard form of this program is

$$\begin{aligned} \text{minimize} \quad & \sup_{u \in U} f_0(x, u) \\ \text{subject to} \quad & \sup_{u \in U} f_i(x, u) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{5.19}$$

In a variant of this approach, we seek to avoid distributional assumptions and instead specify a set of moments for u, such as mean and covariance.

Taking a simple linear program as an example, we start with the underlying problem

$$\begin{aligned} \text{minimize} \quad & c^\mathsf{T} x \\ \text{subject to} \quad & a_i{}^\mathsf{T} x \leq b_i, \quad i = 1, \ldots, L \end{aligned} \tag{5.20}$$

and model uncertainty in the a_i by proposing that a_i was generated by a random process driven by a hidden variable u_i

$$a_i = \bar{a}_i + P_i u, \qquad \|u\| \leq 1 \tag{5.21}$$

where the process has mean $\bar{a}_i$ and covariance matrix $P_i \in \mathbb{R}^{m \times n}$. Substituting these uncertain constraints results in the robust version of the linear program, which turns out to be a

second-order cone program

$$\begin{aligned} &\text{minimize} && c^{\mathsf{T}}x \\ &\text{subject to} && \bar{a_i}^{\mathsf{T}}x + \|P_i{}^{\mathsf{T}}x\| \leq b_i, \quad i = 1, \ldots, L \end{aligned} \tag{5.22}$$

The additional *regularization terms* $\|P_i{}^{\mathsf{T}}x\|$ are typical of robust versions of convex programs: expressing uncertainty in a_i is in some sense equivalent to constraining x to lie away from the directions of greatest uncertainty of the a_i. The quadratic program we describe in Chapter 6 may be seen as a robust optimization problem. The robust linear program developed by [Lanckriet et al. 2002] for text classification – while obtaining approximate performance guarantees on the misclassification probability – is one such example in practical use.

5.3 Convex programming implementations

Convex programming methods are fast approaching 'technology' status. Well-designed, fast, modular libraries such as the Matlab CVX toolkit from Stanford [Boyd & Vandenberghe 2004] and Python CVXOPT [Vandenberghe 2008] are available for high-level program specification and solution. Internally, they make use of a specialized program called *solver* that operates on convex programs in some canonical form. Each solver typically specializes in a particular type of program, such as LP, QP, SOCP, and so on. High-level software such as CVXOPT hides this complexity by automatically selecting the correct solver for the given program.

Current state-of-the-art solvers use a class of algorithm known as interior-point methods for nonlinear convex optimization problems. Examples of current solver software include MOSEK [Anderson 1999] and SeDuMi [Sturm 2004]. Interior-point methods have polynomial-time worst case complexity and can efficiently handle problems involving hundreds of constraints and thousands of variables. As [Boyd & Vandenberghe 2004] point out, in some sense, once the program is specified correctly, it is essentially solved.

Thus, the remaining work in using convex programming methods lies in two areas. First, the problem objectives and constraints must be appropriately analyzed and an appropriate program specified. The objective functions must have some proven connection to reality and desirable outcomes. Second, statistics used in the objective function and con-

straints, such as the expected utility and variance of a document score, must be reasonably estimated and calibrated. Accurate estimation means that we have a method for producing a provably good approximation given the distribution assumption we make. By calibrated, informally we mean that a number such as estimated probability of relevance does have reasonable correlation with actual relevance on average.

For the actual solution of convex programs, our system currently uses the CVXOPT package [Vandenberghe 2008], written in Python. Our main query processing routines in C++ call into CVXOPT via the Python C API. We did not need to make any modifications to CVXOPT itself (as of version 0.9).

While there is little work to date applying convex optimization methods to information retrieval, we note that the use of convex optimization is increasingly becoming used in the database community. Such is the approach proposed by Gibas, Zheng, and Ferhatosmanoglu [Gibas et al. 2007] to use a convex solver to prune computation and I/O overhead for determining a top-*k* ranking of records where the scoring function is convex.

5.4 Conclusions

This chapter introduced the idea of applying convex optimization methods to information retrieval problems. In particular, we focused on providing the background needed for the specific problem of estimating robust query models, which is discussed next in detail in Chapter 6. The use of convex optimization methods gives an efficient way to search a richer space of potential query models, whose quality is evaluated with respect to properties of the whole set, such as a balanced representation of multiple query aspects. This is in contrast to current greedy approaches that only look at a one-dimensional selection process involving a threshold on term score or rank. A convex optimization approach also gives a unifying framework for specifying models that embody competing tradeoffs, such as between choosing terms with high, but uncertain, relevance scores. Finally, by encoding heuristics in the feasible set of query models, we obtain a way to perform selective expansion with multiple criteria.

Chapter 6

Optimization Methods for Query Model Estimation

Our aim in this chapter is to develop a rigorous theoretical basis for automatically estimating reliable expanded query models. We present a novel framework that treats query model estimation as a convex optimization problem. Informally, we seek query models that use a set of terms with high expected relevance but low expected risk. This approach gives a natural way to perform robust selective expansion: if there is no feasible solution to the optimization problem, we do not attempt to expand the original query. Useful additional model constraints such as aspect coverage and cost functions (e.g. to favor short expansions) can be expressed within this framework to give a very flexible general-purpose approach to finding effective query models in a variety of useful retrieval scenarios.

Current methods for calculating query models suffer from several drawbacks. First, existing methods have little principled accounting for the *risk* associated with a particular term. This is partly because a bit of extra computational work must be done to gather the evidence required to make such risk estimates. (In Chapters 3 and 4. we introduced one method, query variants, that is effective in gathering this data efficiently.) As a result, current methods calculate term weights primarily by an expected reward criterion, such as probability in a generative relevance model. Such terms may be high-reward, but also related to few or no other query terms, making them more high risk. The result is generally the unstable feedback results that we see in even state-of-the-art feedback algorithms.

Second, selection of expansion terms is typically done in a greedy fashion by rank or

score, which ignores the properties of the terms as a set and leads to the problem of aspect imbalance, which in turn leads to query drift.

Third, few existing feedback algorithms are able to perform query expansion *selectively*. That is, most query expansion techniques cannot detect when a query is risky and automatically scale down or avoid expansion in such cases. Instead, if query expansion is used at all, it is usually applied to all queries, regardless of risk. A recent study by [Amati et al. 2004] proposed to solve the problem of selective expansion by attempting to predict which queries should have expansion applied or not. They developed a decision criterion based on an ad-hoc combination of heuristics such as query length and rarity of query terms. However, their approach remains an all-or-nothing solution that either keeps or rejects all expansion termsm. Ideally, we want a more flexible algorithm that can determine the best number of expansion terms to use automatically (including possibly none).

Finally, there may be other factors that must be constrained, such as the computational cost of sending many expansion terms to the search engine, or other set-based properties of the expansion terms. To our knowledge such situations are not handled by any single query model estimation framework in a principled way – especially when we must reconcile these competing goals somehow.

To remedy all of these problems, we need a better theoretical framework for query model estimation: one that incorporates both risk and reward data about terms; that detects risky situations and expands selectively by automatically choosing the right number of expansion terms; that can incorporate arbitrary additional problem constraints such as a computational budget; and that has a fast practical implementation.

The central tool that we propose to provide such a risk framework is convex optimization. An optimization approach frees us from the need to provide a closed-form formula for term weighting. Instead, we specify a (convex) objective function and a set of constraints that a good query model should satisfy, letting the solver do the work of searching the space of feasible query models. If no feasible model is found, we do not attempt to expand the original query. Such an approach makes it easy to add custom constraints.

The basic building blocks of our risk framework have two parts. First, we seek to minimize an objective that consists of two criteria: term relevance, and term risk. Term risk considers both the risk of an individual term, and the conditional risk of choosing one term given we have already chosen another. Second, we specify constraints on what ‘good’ sets of terms should look like. These constraints are chosen to address traditional reasons

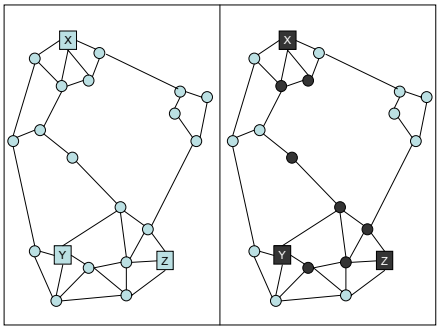


Figure 6.1: Query model estimation as a constrained graph labeling problem using two labels (relevant, non-relevant) on a graph of pairwise term relations. The square nodes X, Y, and Z represent query terms, and circular nodes represent potential expansion terms. Dark nodes represent terms with high estimated label weights that are likely to be added to the initial query. Additional constraints can select sets of terms having desirable properties for stable expansion, such as a bias toward relevant labels related to multiple query terms (right).

for query drift.

This chapter is organized as follows. In Section 6.1 we formulate query term weighting as a graph labeling problem and describe linear and quadratic problems in their basic form. In Section 6.2 we develop the specific objective and constraint functions that will be useful for robust query model estimation, and give the basic complete convex program used for query model estimation. We give examples of how the basic model may be refined in Section 6.3.1, including how to implement a non-convex budget constraint. We demonstrate the effectiveness of our convex formulation in Section 6.4 on standard test collections and explore the contributions of each constraint type on the quality of the estimated query model. Section 6.5 is a discussion of some implications of our work for query expansion and Section 6.6 discusses related work.

6.1 Query model estimation as a graph labeling problem

We can gain some insight into the problem of query model estimation by viewing the process of building a query as a two-class *labeling* problem over terms. Given a vocabulary V, for each term $t \in V$ we decide to either add term t to the query (assign label ‘1’ to the term), or to leave it out (assign label ‘0’). The initial query terms are given a label of ‘1’.

Our goal is to find a function $f : V \rightarrow \{0, 1\}$ that classifies the finite set V of $|V| = K$ terms, choosing one of the two labels for each term. The terms are typically related, so that the pairwise similarity $\sigma(i, j)$ between any two terms w_i, w_j is represented by the weight of the edge connecting w_i and w_j in the undirected graph $G = (V, E)$, where E is the set of all edges. The cost function $L(f)$ captures our displeasure for a given f, according to how badly the following two criteria are given by the labeling produced by f.

- The cost $c_{i:k}$ gives the cost of labeling term t_i with label $k \in \{0, 1\}$.
- The cost $\sigma_{i,j} \cdot d(f(i), f(j))$ gives the penalty for assigning labels $f(i)$ and $f(j)$ to items i and j when their similarity is $\sigma_{i,j}$. The function $d(u, v)$ is a metric that is the same for all edges. Typically, similar items are expected to have similar labels and thus a penalty is assigned to the degree this expectation is violated.

For this study, we assume a very simple metric in which $d(i, j) = 1$ if $i \neq j$ and 0 otherwise. In a probabilistic setting, finding the most probable labeling can be viewed as a form of maximum a posteriori (MAP) estimation over the Markov random field defined by the term graph.

Although this problem is NP-hard for arbitrary configurations, various approximation algorithms exist that run in polynomial time by relaxing the constraints. Here we relax the condition that the labels be integers in $\{0, 1\}$ and allow real values in $[0, 1]$. A review of relaxations for the more general metric labeling problem is given by Ravikumar and Lafferty [Ravikumar & Lafferty 2006]. The basic relaxation we use is

$$\begin{aligned} \text{maximize} \quad & \sum_{s;j} c_{s;j} x_{s;j} + \sum_{s,t;j,k} \sigma_{s,j;t,k} x_{s;j} x_{t;k} \\ \text{subject to} \quad & \sum_{j} x_{s;j} = 1 \\ & 0 \leq x_{s;j} \leq 1 \end{aligned} \tag{6.1}$$

The variable $x_{s;j}$ denotes the assignment value of label j for term s. For a two-class problem where $j \in \{0, 1\}$, the values of x for one class completely determine the values for the other class since they must sum to 1. It therefore suffices to optimize over only the x_s for one class, and to simplify matters, we often refer to c_s or $\sigma_{s,t}$ instead of $c_{s;j}$ or $\sigma_{s,j;t,k}$.

Our method obtains its initial assignment costs $c_{s;j}$ based on term weights from a baseline feedback method, given an observed query and corresponding set of query-ranked

documents. For our baseline expansion method, we use the strong default feedback algorithm included in Indri 2.2 based on Lavrenko's Relevance Model [Lavrenko 2004]. Further details are available in [Collins-Thompson & Callan 2007].

In the next section, we discuss how to specify values for $c_{s;j}$ and $\sigma_{s,j;t,k}$ that make sense for query model estimation. Our goal is to find a set of weights $x = (x_1, \ldots, x_K)$ where each x_i corresponds to the weight in the final query model of term w_i and thus is the relative value of each word in the expanded query. The graph labeling formulation may be interpreted as combining two natural objectives: the first maximizes the expected relevance of the selected terms, and the second minimizes the risk associated with the selection. We now describe each of these in more detail, followed by a description of additional set-based constraints that are useful for query expansion.

6.2 Objectives and constraints for query model estimation

Typically, our optimization will embody a basic tradeoff between wanting to use evidence that has strong expected relevance, such as expansion terms with high relevance model weights, and the risk or confidence in using that evidence. We begin by describing the objectives and constraints over term sets that might be of interest for estimating query models, and then show how these properties can be formulated as (sometimes competing) constraints or objectives in a convex optimization problem. The object of this section is not to describe a specific strategy for formulating a query, but rather to address the problem of how to weight the different sources of evidence (words) that the query will be based on.

6.2.1 Relevance objectives

Given an initial set of term weights from a baseline expansion method $c = (c_1, \ldots, c_K)$ the *expected relevance* over the vocabulary V of a solution x is given by the weighted sum $c \cdot x = \sum_k c_k x_k$. Essentially, maximizing expected relevance biases the 'relevant' labels toward those words with the highest c_i values. Other relevance objective functions are also possible, as long as they are convex. For example, if c and x represent probability distributions over terms, then we could replace $c \cdot x$ with $KL(c||x)$ as an objective since KL-divergence is also convex in c and x.

The initial assignment costs (label values) c can be set using a number of methods depending on how scores from the baseline expansion model are normalized. In the case of Indri's language model-based expansion, we are given estimates of the Relevance Model

$p(w|R)$ over the highest-ranking k documents[1]. We can also estimate a non-relevance model $p(w|N)$ using the collection to approximate non-relevant documents, or using the *lowest-ranked k* documents out of the top 1000 retrieved by the initial query Q. To set $c_{s:1}$, we first compute $p(R \mid w)$ for each word w via Bayes Theorem,

$$p(R|w) = \frac{p(w|R)}{p(w|R) + p(w|N)} \tag{6.2}$$

assuming $p(R) = p(N) = 1/2$. Using the notation $p(R|Q)$ and $p(R|\overline{Q})$ to denote our belief that any query word or non-query word respectively should have label 1, the initial expected label value is then

$$c_{s:1} = \begin{cases} p(R|Q) + (1 - p(R|Q)) \cdot p(R|w_s) & s \in Q \\ p(R|\overline{Q}) \cdot p(R|w_s) & s \notin Q \end{cases} \tag{6.3}$$

for the 'relevant' label. We use $p(R|Q) = 0.75$ and $p(R|\overline{Q}) = 0.5$. Since the label values must sum to one, for binary labels we have $c_{s:0} = 1 - c_{s:1}$. It may be possible to use more sophisticated methods for setting the $c_{s;j}$ such as different probability distribution models of relevance and non-relevance scores.

6.2.2 Risk objectives

Optimizing for expected term relevance only considers one dimension of the problem. A second critical objective is minimizing the risk associated with a particular term labeling. We adapt an informal definition of risk here in which the variance of the expected relevance is a proxy for uncertainty, encoded in the matrix Σ with entries σ_{ij}. Using a betting analogy, the weights $x = \{x_i\}$ represent wagers on the utility of the query model terms. A risky strategy would place all bets on the single term with highest relevance score. A lower-risk strategy would distribute bets among terms that had both a large estimated relevance and low redundancy, to cover all aspects of the query.

Conditional term risk. First, we consider the *conditional risk* σ_{ij} between pairs of terms w_i and w_j. To quantify conditional risk, we measure the redundancy of choosing word w_i given that w_j has already been selected. This relation is expressed by choosing a symmetric similarity measure $\sigma(w_i, w_j)$ between w_i and w_j, which is rescaled into a distance-like

[1] We use the symbols R and N to represent relevance and non-relevance respectively.

measure $d(w_i, w_j)$ with the formula

$$\sigma_{ij} = d(w_i, w_j) = \gamma \exp(-\rho \cdot \sigma(w_i, w_j)) \tag{6.4}$$

The quantities γ and ρ are scaling constants that depend on the output scale of σ, and the choice of γ also controls the relative importance of individual vs. conditional term risk.

In this study, the $\sigma(w_i, w_j)$ measure is the perturbation kernel defined in Chapter 4. Details on the perturbation kernel parameters used for evaluation are given in Section 6.4.1.

Individual risk. We say that a term related to multiple query terms exhibits *term centrality*. Previous work has shown that central terms are more likely to be more effective for expansion than terms related to few query terms [Collins-Thompson & Callan 2005] [Xu & Croft 1996]. We use term centrality to quantify a term's individual risk, and define it for a term w_i in terms of the vector d_i of all similarities of w_i with all query terms. The covariance matrix Σ then has diagonal entries

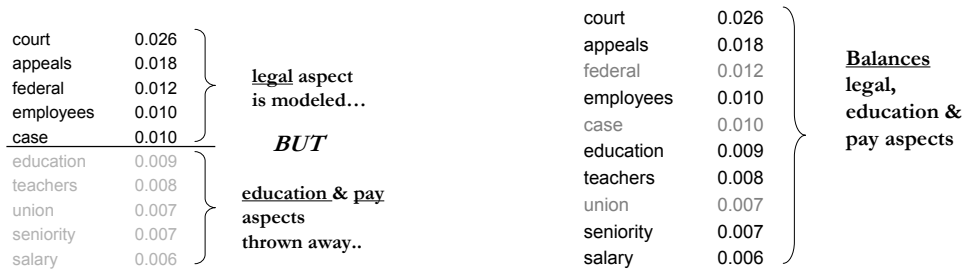
$$\sigma_{ii} = \|d_i\|_2^2 = \sum_{w_q \in Q} d^2(w_i, w_q) \tag{6.5}$$

Other definitions of centrality are certainly possible, e.g. depending on generative assumptions for term distributions.

We can now combine relevance and risk into a single objective, and control the tradeoff with a single parameter κ, by minimizing the function

$$L(x) = -c^T x + \frac{\kappa}{2} x^T \Sigma x. \tag{6.6}$$

If Σ is estimated from term co-occurrence data in the top-retrieved documents, then the condition to minimize $x^T \Sigma x$ also encodes the fact that we want to select expansion terms that are not all in the same co-occurrence cluster. Rather, we prefer a set of expansion terms that are more diverse, covering a larger range of potential topics. Risk estimates may also come from the similarity measures given by data perturbation kernels described in Chapter 4. Looking beyond criteria that focus on individual terms or pairs of terms, we now discuss *set-based constraints*, which are properties of the entire set of expansion terms.



(a) Greedy term selection

(b) Set-based term selection

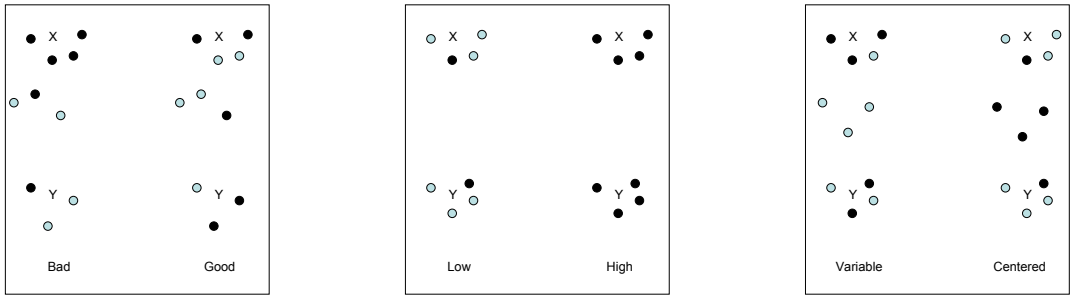
Figure 6.2: Hypothetical query: Merit pay law for teachers, Showing greedy expansion term selection (left) and set-based selection (right)

6.2.3 Set-based constraints

One restriction of current query enhancement methods is that they typically make term-by-term decisions, without considering the qualities of the *set* of terms as a whole. An example of this behavior is shown in Figure 6.2. We can identify three broad, complementary aspects for a query like *merit pay law for teachers*: a legal aspect, a financial aspect, and an educational aspect[2]. A one-dimensional greedy selection by term score, especially for a small number of terms, has the risk of emphasizing terms related to one aspect, such as *law*, and not others. This in turn increases the risk of query drift. A more stable feedback algorithm would select terms that cover all three aspects of the query. We call this property *aspect balance*. Figure 6.3 gives a graphical example of aspect balance (left), along with another criterion, which we call *aspect coverage* (center), and the term centrality objective (right) given earlier.

In this figure, each subfigure shows a different constraint. For each constraint, two possible colorings of a word graph are shown on the left and right. The word graph contains words related to two hypothetical query terms *X* and *Y*. The dots of the graph represent the related words (vertices) – the edges and word labels of the graph have been omitted for clarity, to focus on the nature of the labeling. Dots colored black are words selected to form the query expansion for *X* and *Y*. Conversely, light-colored dots are words that are *not* included in the expansion. The left-hand labeling in a subfigure selects a subset of terms that does a weak job of satisfying the constraint (aspect balance, aspect coverage, or

[2] Alternative aspect lists could certainly be defined for this example, such as a more generic *location, procedure, time*. The important point is that any aspect that covers the meaning of the query should have a similar covering in the expanded version of the query.



(a) Aspect balance (b) Aspect coverage (c) Term centering

Figure 6.3: Three complementary criteria for expansion term weighting on a graph of candidate terms, and two query terms X and Y. The aspect balance constraint (Subfig. 6.3a) prefers sets of expansion terms that balance the representation of X and Y. The aspect coverage constraint (Subfig. 6.3b) increases recall by allowing more expansion candidates within a distance threshold of each term. Term centering (Subfig. 6.3c) prefers terms near the center of the graph, and thus more likely to be related to both terms, with minimum variation in the distances to X and Y.

term centering). The right-hand labeling selects a subset of terms that strongly satisfies the constraint. Note that some coloring can satisfy some of these constraints but not others. For example, the term subset selected in the left-hand labeling of Figure 6.3(c) has good aspect balance and coverage, but is not strongly centered between the original query terms X and Y.

We now define aspect balance and other set-based constraints more formally. To define these constraints mathematically, we map terms to their co-ordinates in the data perturbation space defined in Chapter 4. In this space, each word w_i has a corresponding feature vector $\phi(w)$ with entries $\phi_k(w)$

$$\phi_k(w) = \frac{\sqrt{p(w|\theta_k)} - \sqrt{p(w|\theta_{\boldsymbol{q}})}}{\sqrt{p(w|\theta_{\boldsymbol{q}})}}. \tag{6.7}$$

The similarity between words becomes a simple Euclidean distance in this space. Details on the specific settings for the perturbation kernel used in this chapter are given in Section 6.4.1. We now show how to write these constraints as optimization constraints in terms of these feature vectors.

Aspect balance. Recall that in Chapter 4 we created the matrix A from the vectors $\phi(w_i)$ of perturbations for each word w_i, i.e.

$$\phi_k(w_i) = A_{ik} = p(w_i|q_{(\alpha_k)}) \tag{6.8}$$

where A_{ik} is the (i,k)-th entry of A. In this chapter we restrict A to just the query term feature vectors $\phi(\boldsymbol{q}_i)$, making the simplistic assumption that each of a query's terms represents a separate and unique aspect of the user's information need. We create the matrix A from the vectors $\phi(w_i)$ where $\phi_k(w_i) = \sigma_{ik}$ for each query term $\boldsymbol{q}_k$. (Recall that σ_{ik} is the i,k-th entry in matrix Σ given by Eq. 6.4.) In effect, Ax gives the projection of the solution model x on each query term's feature vector $\phi(\boldsymbol{q}_i)$. The requirement that x be in balance is equivalent to the requirement that the mean of the projections be equal to the mean ζ_μ of the $\phi(\boldsymbol{q}_i)$. This is expressed as

$$Ax \leq \mu + \zeta_\mu \tag{6.9}$$

To demand an exact solution, we set $\zeta_\mu = 0$. In reality, some slack is desirable for slightly better results and so we use a small positive value such as $\zeta_\mu = 2.0$.

The assumption that each query term and the perturbation features associated with it represent a different aspect of the information need is somewhat unrealistic, but it greatly simplifies the model. In a more general Bayesian treatment, aspects would be described by latent variables, which in turn would require treating the matrix A as uncertain, resulting in a second-order cone optimization problem[3].

Query term support. Another important constraint is that the set of initial query terms Q be predicted by the solution labeling. We express this mathematically by requiring that the the weights for the 'relevant' label on the query terms $x_{i:1}$ lie in a range $\beta_i \leq x_i \leq u_i$ and in particular be above the threshold β_i for $x_i \in Q$. Currently β_i is set to a default value of 0.95 for all query terms, and zero for all other terms. u_i is set to 1.0 for all terms. Term-specific values for β_i may also be desirable to reflect the rarity or ambiguity of individual words.

Aspect coverage. One of the strengths of query expansion is its potential for solving the vocabulary mismatch problem by finding different words to express the same information need. Therefore, we can also require a minimal level of *aspect coverage*. That is, we

[3]We actually did implement this generalization, but encountered technical problems with the current solver in obtaining stable solutions in higher dimensions. We expect to pursue this in future work.

$$\begin{aligned}
&\text{minimize} && -c^{\mathsf{T}}x - \frac{\kappa}{2}x^{\mathsf{T}}\Sigma x && \textit{Relevance, term centrality (risk)} && (6.12)\\
&\text{subject to} && Ax \le \mu + \zeta_{\mu} && \textit{Aspect balance} && (6.13)\\
& && g_i{}^{\mathsf{T}}x \ge \zeta_i, \quad w_i \in Q && \textit{Aspect coverage} && (6.14)\\
& && \beta_i \le x_{i;1} \le u_i, \quad w_i \in Q && \textit{Minimum query term support} && (6.15)\\
& && \sum_k x_{i;k} = 1 && \textit{Label uniqueness} && (6.16)\\
& && x \ge 0 && \textit{Positivity} && (6.17)
\end{aligned}$$

Figure 6.4: The basic quadratic program QMOD used for query model estimation.

may require more than just that terms are balanced evenly among all query terms: we may care about the absolute level of support that exists. For example, suppose our information sources are feedback terms, and we have two possible term weightings that are otherwise feasible solutions. The first weighting has only enough terms selected to give a minimal non-zero but even covering to all aspects. The second weighting scheme has three times as many terms, but also gives an even covering. Assuming no conflicting constraints such as maximum query length, we may prefer the second weighting because it increases the chance we find the right alternate words for the query, potentially improving recall.

We denote the vector of distances from query term $\boldsymbol{q}_i$ in query Q to neighboring words by the vector g_i, which has entries g_{ik} defined by

$$g_{ik} = d(w_i, w_k) \qquad k = 1 \ldots K, \quad w_i \in Q \tag{6.10}$$

where $d(w_i, w_k)$ is given in Eq. 6.4. The projection ${g_i}^T x$ gives us the aspect coverage, or how well the words selected by the solution x 'cover' term $\boldsymbol{q}_i$. The more expansion terms near $\boldsymbol{q}_i$ that are given higher weights, the larger this value becomes. When only the query term is covered, the value of ${g_i}^T x = \sigma_{ii}$. We want the aspect coverage for each of the vectors g_i to exceed a threshold ζ_i, and this is expressed by the constraint

$${g_i}^T x \ge \zeta_i. \tag{6.11}$$

6.2.4 Combining objectives and constraints

Putting together the relevance and risk objectives, and constraining by the set properties, results in the following complete quadratic program for query model estimation, which we call QMOD, shown in Figure 6.4. The purpose of each objective or constraint is given in italics on the right. The roles and default values of the various QMOD parameters are summarized in Table 6.1. Sample output from running QMOD is shown in Figure 6.5. Note that a term like *disorders* may have a much higher initial relevance weight than another term (e.g. *brain*), but because *disorders* is highly redundant with the other key terms *syndrome* and *disease*, the term *disorders* is actually removed from the solution, while *brain* is retained.

We perform an extensive evaluation of the sensitivity of retrieval performance to these parameters across multiple collections in Section 6.4.3. Fig. 6.9 shows how a typical solution changes as the objective parameter κ increases, moving from focusing on just the highest relevance weights, to a more diverse set of terms.

6.3 Extensions to the basic model

We now show how extra criteria, such as budget constraints, weight diversification, and uncertainty in the relevance parameters are easy to add to the basic QMOD framework.

6.3.1 Budget constraints

We can specify there is a computation cost ϵ for each term w in a query we send to the server. If our goal is to find the optimal language model , we can add a constraint that the total query cost must be less than ϵ_{Max}. The optimal query will then consist of a small subset of terms, each of which covaries significantly with multiple important terms in the document. In most realistic scenarios, the cost of adding another expansion term can be approximated as fixed at a constant value β: no term is much more expensive to add to a query than any other, and the total cost is simply linear in the number of query terms[4]. Because of the non-convex nature of the fixed cost constraint (zero at zero, non-zero constant everywhere else) this cannot be solved directly by convex optimization, but there are heuristics that allow us to get close to optimal quickly.

[4]Of course, more sophisticated cost functions also fit within this framework, in the case where increase in cost is greater than linear.

Symbol	Name	Meaning	Default
κ	Bi-objective tradeoff	Controls tradeoff between the influence of initial relevance assignment costs and term covariance. Higher κ means more focus on the term covariance objective.	1.0
ζ_μ	Balance tolerance	Controls tightness of balance among all query aspects. Higher ζ_μ relaxes the balance requirement between all query aspects.	2.0
ζ_i	Aspect coverage	Controls the minimum support for each query aspect. Higher ζ_i requires more expansion terms near each query term. When $\zeta_i = 0$, no expansion terms are strictly required: the query term alone is sufficient. As ζ_i increases, the feasible set shrinks and solutions become more conservative.	0.1
γ	Conditional covariance	Higher γ gives more influence to conditional risk (term covariance) compared to individual term risk. For example, if $\gamma = 0$ then only individual term risk values (the diagonal of the covariance matrix Σ) are used.	0.75
β_i	Minimum query term support	Minimum final label value for each query term. Reducing β_i for the i-th query term $\boldsymbol{q}_i$ relaxes the requirement that $\boldsymbol{q}_i$ must be predicted by the query model.	0.95

Table 6.1: Summary of control parameters for basic QMOD quadratic program.

```
Initial assignment weights:
parkinson:  0.996
disease:  0.848
syndrome:  0.495
patients:  0.313
parkinsons:  0.492
brain:  0.360
disorders:  0.491
treatment:  0.289
patient:  0.483
diseases:  0.153
...
pcost dcost gap pres dres
0:  0.0000e+00 3.8908e+00 5e+01 1e+00 1e+00
1:  -1.5007e+00 2.1905e+01 4e+01 9e-01 9e-01
2:  2.5299e+00 4.5050e+01 4e+01 1e-00 7e-01
3:  1.8112e+01 6.5871e+01 3e+01 1e+00 4e-01
...
13:  8.8980e+01 8.8980e+01 7e-05 2e-06 1e-14
14:  8.8980e+01 8.8980e+01 2e-06 4e-08 1e-14
(Successful convergence of primal and dual solutions.)
Final label values:
...
parkinson:0.9900
disease:0.9900
syndrome:0.2077
patients:0.0918
parkinsons:0.1350
brain:0.0256
disorders:0.0000
treatment:0.0000
patient:0.0000
diseases:0.0000
...
```

Figure 6.5: Excerpts from output of CVXOPT solver on a constrained quadratic program, showing elements of the x solution vector (final label values). The query in this example is TREC topic 454: “parkinson’s disease”.

One heuristic is to simply set a threshold on the x_i and set any x_i below the threshold to zero. A more sophisticated but also more principled approach recently described by [Lobo et al. 2007] uses a convex relaxation of the fixed cost constraint that utilizes estimates of upper and lower bounds on the values of the x_i. If we can compute upper and lower bounds on the x_i, we can use the initial non-sparse solution as input to a regularization step, where we find a sparse vector y that minimizes the $\ell 1$-norm distance to the original solution, with the non-zero y_i falling within the corresponding x_i bounds.

We denote the cost function for a solution (query weights) x by $\phi(x)$ which gives the general query cost constraint

$$\mathbf{1}^T x + \phi(x) \leq 0 \tag{6.18}$$

If $\phi(x)$ is a convex function, this constraint defines a convex set. We can define individual costs for each element of x by writing

$$\phi(x) = \sum_{i=1}^{n} \phi_i(x) \tag{6.19}$$

where ϕ_i is the cost function for element i. In the simplest case, there are no costs associated with any query term[5], and so $\phi(x) = 0$. If there is no use of element x_i in the solution, $\phi_i = 0$. Portfolio optimization considers a general fixed-plus-linear cost function in which costs are a linear function of the x_i, this makes much less sense for information retrieval. If the x_i represent query term weights, we do not expect a query to run more slowly as one term's weight is increased over another term's (non-zero) weight[6]. However, the specific case of a *fixed cost* for an x_i is very relevant to retrieval, because adding more terms to a query *does* increase processing costs. We therefore focus on the specific case where $\phi(x_i) = \beta$ for some constant β.

We adapt a heuristic introduced in the thesis of Fazel [Fazel 2002] and recently first applied to portfolio optimization in [Lobo et al. 2007]. We solve successive refinements of QMOD such that at the k-th step we use a cost function $\phi^k(x) = \sum_i \phi_i^{(k)}(x_i)$. in the cost

[5] Or any other information source. For specificity we work with information sources being terms.

[6] Different terms, however, might indeed have different processing costs when the weight is non-zero. For example, adding a word with a huge term frequency could slow down a query much more than adding a very rare term, due to the time required to process inverted lists from the index.

constraint [7] by using iterative refinements of ϕ:

$$\phi_i^{(k)}(x_i) = \left(\frac{\beta_i}{|x_i^{(k)}| + \delta}\right)|x_i| \tag{6.20}$$

where β_i is the fixed cost associated with x_i, and δ is a small constant that acts like a threshold, below which any value is deemed to be zero. Intuitively, this heuristic successively increases the cost of small x_i as they become smaller, thus accelerating them toward zero, while satisfying the other constraints of the problem. Further details such as proof of convergence of this heuristic are given in [Lobo et al. 2007].

$$\phi_i(x_i) = \begin{cases} 0 & x_i = 0 \\ \beta & x_i \neq 0 \end{cases} \tag{6.21}$$

The above heuristic may be overly general for most practical retrieval scenarios, since x_i can never be negative and computation cost does not generally increase as a function of the term weight *magnitude*. However, it may be desirable to set different *fixed* costs for each term that reflect factors such term frequency, since term frequency affects the size of inverted list that must be loaded at search time. In these simpler cases, it would suffice to use a vector of penalty weights within an ℓ-1-norm constraint that controls sparsity.

6.3.2 Weight diversification

Another useful type of constraint is to specify that no more than a fraction $\alpha, \alpha \in [0, 1]$ of the total weight mass be allocated to the largest r term weights. This can be encoded as follows (see [Boyd & Vandenberghe 2004], 5.19).

$$\begin{aligned} \text{maximize} \quad & c^\mathsf{T}x - \frac{\kappa}{2}x^\mathsf{T}\Sigma x \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T}x = 1, \quad x \geq 0 && (6.22) \\ & r \cdot t + \sum_k u_k \leq \alpha && (6.23) \\ & \mathbf{1}^\mathsf{T}t + u \geq x && (6.24) \\ & u \geq 0 \end{aligned}$$

[7] The heuristic can also be used when the cost function is used in the objective function.

6.3.3 Uncertainty in parameters or objectives

Our estimates of term relevance and risk weights (expected utility and risk) may themselves be uncertain. We might have a range of values for each, or multiple strategies. In this case, we might want to optimize the worst-case scenario given our set of of strategies. We can handle uncertainty in the statistical model for (p, Σ) using multiple such models (p_k, Σ_k).

Interestingly, we can express a constraint of the form $Prob(r \leq \alpha) \leq \beta$ where α is a given unwanted return level and β is a given maximum probability. Or multiple such constraints for various levels of loss.

We can maximize the minimum of the expected returns (conservative) and so combining these into one objective, we have

$$\begin{aligned} &\text{maximize} && \min_k {p_k}^\mathsf{T} x \\ &\text{subject to} && {p_k}^\mathsf{T} x + \Phi^{-1}(\beta)\|\sum_k^{1/2} x\| \geq \alpha \end{aligned} \tag{6.25}$$

We can take our approach one step further and assume there is uncertainty about the aspect matrix A. Using a *stochastic* approach to robust approximation we assume the matrix A is a random variable with mean $\overline{A}$ and noise described by the matrix U, which has first and second moments $\mathbb{E}[U] = 0$ and $\mathbb{E}\left[U^\mathsf{T}U\right] = P$. We minimize $\mathbb{E}\left[\|(\overline{A} + U)x - b\|^2\right]$ (Reminder: we assume $\|\cdot\|_2$ is the default norm denoted by $\|\cdot\|$.)

$$\begin{aligned} \mathbb{E}\left[\|Ax - b\|^2\right] &= \mathbb{E}\left[\|\overline{A}x - b + Ux\|^2\right] && (6.26)\\ &= \|\overline{A}x - b\|^2 + \mathbb{E}\left[x^\mathsf{T}U^\mathsf{T}Ux\right] && (6.27)\\ &= \|\overline{A}x - b\|^2 + x^\mathsf{T}Px && (6.28)\\ &= \|\overline{A}x - b\|^2 + \|P^{1/2}x\|^2 && (6.29) \end{aligned}$$

When $P = \gamma \boldsymbol{I}$ this becomes

$$\|\overline{A}x - b\|^2 + \gamma\|x\|^2 \tag{6.30}$$

This can also be seen as a form of Tikhonov regularization ([Boyd & Vandenberghe 2004],

p. 306). We can use this derivation to obtain a SOCP for relaxing the balance constraint with uncertain A, to implement true latent aspects.

6.3.4 Incorporating query difficulty statistics

A query difficulty statistic $D(\boldsymbol{q})$ amounts to an estimate of a query's likely utility. This may affect our estimate of the expected return of the original query, making feedback more or less attractive. For example, if we had a reliable way to predict that the initial query's performance was likely to be poor, then then we have little to lose by applying query expansion anyway. In this scenario, poor performance after query expansion is not necessarily worse than poor performance before query expansion, which is our implicit assumption when query difficulty is ignored.

In the extreme case where the original retrieval is highly unlikely to be much above zero average precision, the optimal expansion strategy may be affected. For example, we may wish to distribute risk among clusters, choosing one strong representative per cluster – minimizing redundant non-relevant documents and optimizing for recall. The existence of a reliable $D(\boldsymbol{q})$ function, even if the reliability is limited to low or high extremes, also leaves open the possibility of using $D(\boldsymbol{q}_i)$ as a search objective over multiple iterations, finding the maximum $D(\boldsymbol{q}_i)$ for different expanded query candidates $\boldsymbol{q}_i$, or using $D(\boldsymbol{q}_i)$ for improved model combination.

6.4 Evaluation

Our evaluation of the QMOD algorithm has several parts. After describing the details of experimental setup in Section 6.4.1, we confirm in Section 6.4.2 the superior quality of the solutions found by QMOD compared to a strong baseline expansion model. We do this by comparing the risk-reward curves for both methods as the interpolation value α with the initial query is varied from 0 to 1. We also summarize results using standard retrieval measures. It turns out that for most topic sets in the evaluation, the expansions computed by QMOD offer substantial reductions in downside risk, while maintaining or exceeding the precision of the baseline method. Second, in Section 6.4.3 we explore how the QMOD program achieves its robust solutions by analyzing how changes in the objective and constraint parameters (summarized in Table 6.1) affect the risk-reward trade-off curve. Third, in Sections 6.4.4 and 6.4.5 we test the generality of QMOD by applying it to two alternative baseline algorithms. Respectively, these are a Rocchio-type method, and a

very noisy *idf*-only method whose purpose is to verify that QMOD successfully attenuates noise. Finally, in Section 6.4.6 we verify that the feasible set for the QMOD convex program is well-calibrated for the task of selective query expansion. We show that QMOD's constraints focus on solutions for queries with moderate to good expansion performance, while tending to avoid queries at the extreme ends of the performance spectrum.

6.4.1 Evaluation setup

The basic methodology and collections of our evaluation setup are the same as used to evaluate heuristic model combination in Chapter 3. To review, we evaluated performance on six TREC topic sets, covering a total of 700 unique queries. These TREC topic sets are TREC 1&2, TREC 7, TREC 8, wt10g, robust2004, and gov2. Details on TREC topic sets, collections and methodology are given in Appendix C. The same indices, indexing process, and baseline feedback algorithm (Indri) were used as in the evaluation in Section 3.4.1.

As with our experiments in Section 3.1.4, for an n-word query, we used leave-one-out (LOO) query variation to produce $n + 1$ subqueries (query variants), including the entire initial query as one variant.[8] To compute each of the corresponding $n + 1$ feedback models (one for each variant's result set), we also used the same document resampling method and parameters as in Section 3.1.4, i.e. using the top 50 ranked documents, weighting by relevance score, and using 30 document-set resamplings, which were combined by fitting the maximum-likelihood Dirichlet distribution and taking the mode.

We then set the inputs to QMOD as follows. For efficiency, we limited our vocabulary V to the top $n = 100$ expansion term candidates according to the raw expansion score from the Indri algorithm. With these Indri term scores, the entries of the assignment cost vector c were set using Eq. 6.3 in Section 6.2.1. The matrices Σ, A, and g_i were also calculated dynamically for each query using the definitions given in Section 6.2.3. The entries of Σ and g_i are determined by the definition of the distance function $d(w_s, w_t)$, which in turn is defined in terms of the similarity function $\sigma(w_s, w_t)$. In this evaluation, the function $\sigma(w_s, w_t)$ is the perturbation kernel described in Chapter 4. Because it is not possible to calculate the perturbation kernel for single-word queries, the Jaccard method given in Appendix A was used to compute $\sigma(w_s, w_t)$ for single-word queries instead of the perturbation kernel. The matrix A is a skinny $|Q| \times K$ matrix, with each row being the feature vector $\phi(\boldsymbol{q}_i)$ for query

[8] For one-word queries, this means only the original query was used, and the final feedback model was based on document resampling only.

term $\boldsymbol{q}_i$. There is one matrix g_i for each query term $\boldsymbol{q}_i \in Q$, as defined by Eq. 6.10.

To compute the perturbation kernel and feature vectors $\phi(w_i)$, we used the $n + 1$ LOO feedback models calculated above to compute the feature vector $\phi(w_i)$ for each word using Eq. 6.7. In that equation, the value $p(w|\theta_k)$ is the probability of word w in the k-th feedback model ($k = 1 \ldots n$), and $p(w|\theta_q)$ is the feedback model for the initial query $\boldsymbol{q}$. We set the rescaling parameters $\gamma = 0.75$ and $\rho = 0.25$ in Eq. 6.4.

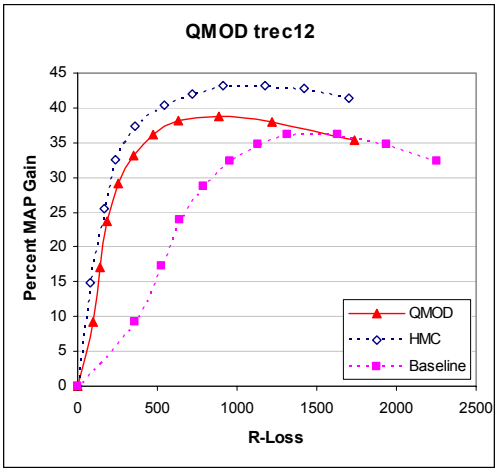
The default values for κ, γ, and the other QMOD control parameters were set to the values shown in Table 6.1 for all collections. We obtained these collection-wide parameter settings empirically: to obtain realistic generalization, we first examined performance variation over TREC 1&2, TREC 7, and TREC 8. We then picked a common set of parameters that gave consistent performance across those collections. After these parameters and the optimization code were frozen, we ran experiments on the other three collections: wt10g, robust2004, and gov2. We used a maximum of 100 iterations when running QMOD, and used the default convergence settings for the cvxopt QP solver.

After the QMOD program was run, we chose the top 20 terms according to soft label value (i.e. the solution values found by QMOD). Terms with a soft label value of less than 0.01 were ignored. Some queries had less than 20 final expansion terms with non-zero weight, either because the QMOD program was infeasible – resulting in no expansion terms at all – or simply because of the sparsity-seeking nature of the objective. Finally, to combine the feedback model output by QMOD with the initial query, we used the same default interpolation setting $\alpha = 0.5$ as the experiments in Chapter 3 (except where otherwise indicated to produce tradeoff curves).

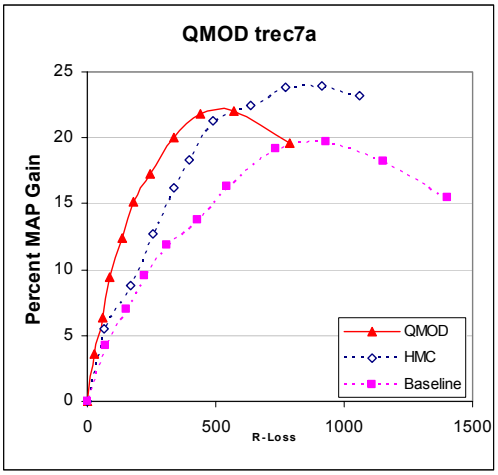
6.4.2 Risk-reward performance

In this section we evaluate the robustness of the query models estimated using the convex program in Fig. 6.4 over multiple standard TREC collections. As we noted earlier in Chapter 1, there are elements of both risk and reward in attempting to find effective, robust query models. Our primary tool for summarizing the trade-off in these two objectives is the is the *risk-reward curve*, and the *robustness histogram*.

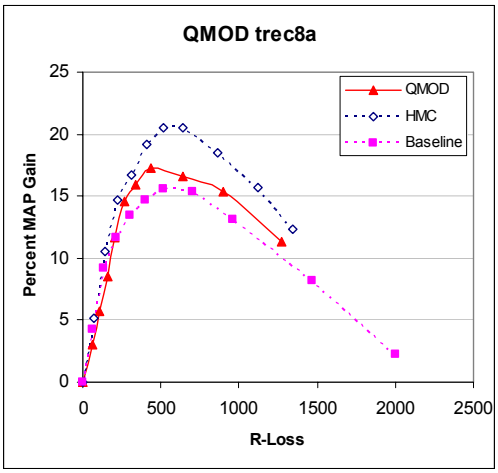
Risk-reward curves Risk-reward curves for six TREC topic sets are shown in Figure 6.6. The x-axis summarizes downside risk with R-Loss, the net loss in relevant documents lost due to expansion failures. The y-axis summarizes reward using percentage MAP gain over the initial query (no expansion). The solid (red) line is the curve given by the robust QMOD



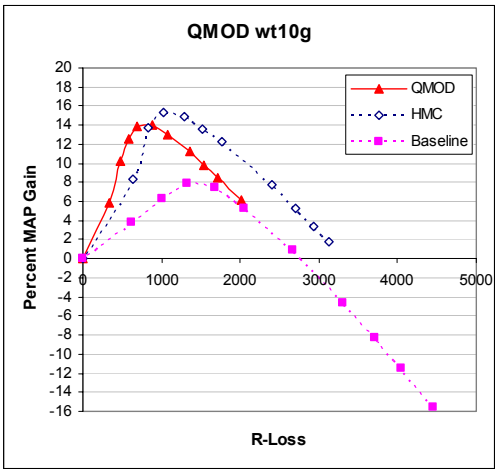
(a) TREC 1& 2



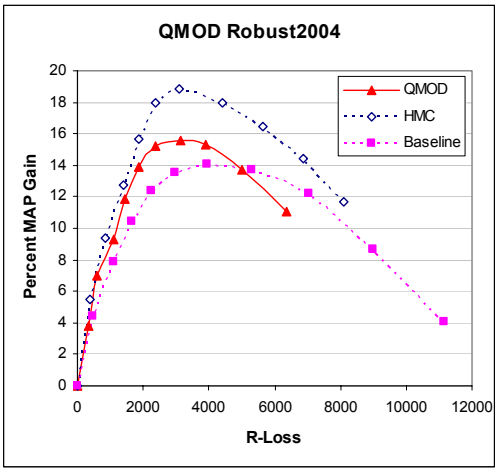
(b) TREC 7



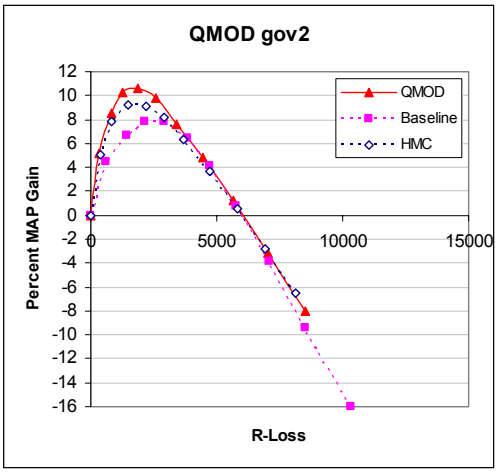
(c) TREC 8



(d) wt10g



(e) Robust 2004



(f) gov2

Figure 6.6: Risk-reward tradeoff curves for six TREC topic sets, showing how the QMOD and HMC robust feedback methods consistently dominate the performance of the baseline feedback method. HMC is the heuristic model combination method from Chap. 3. The baseline feedback model is the Indri Relevance Model. Tradeoff curves that are *higher and to the left* are better. Points are plotted in α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.

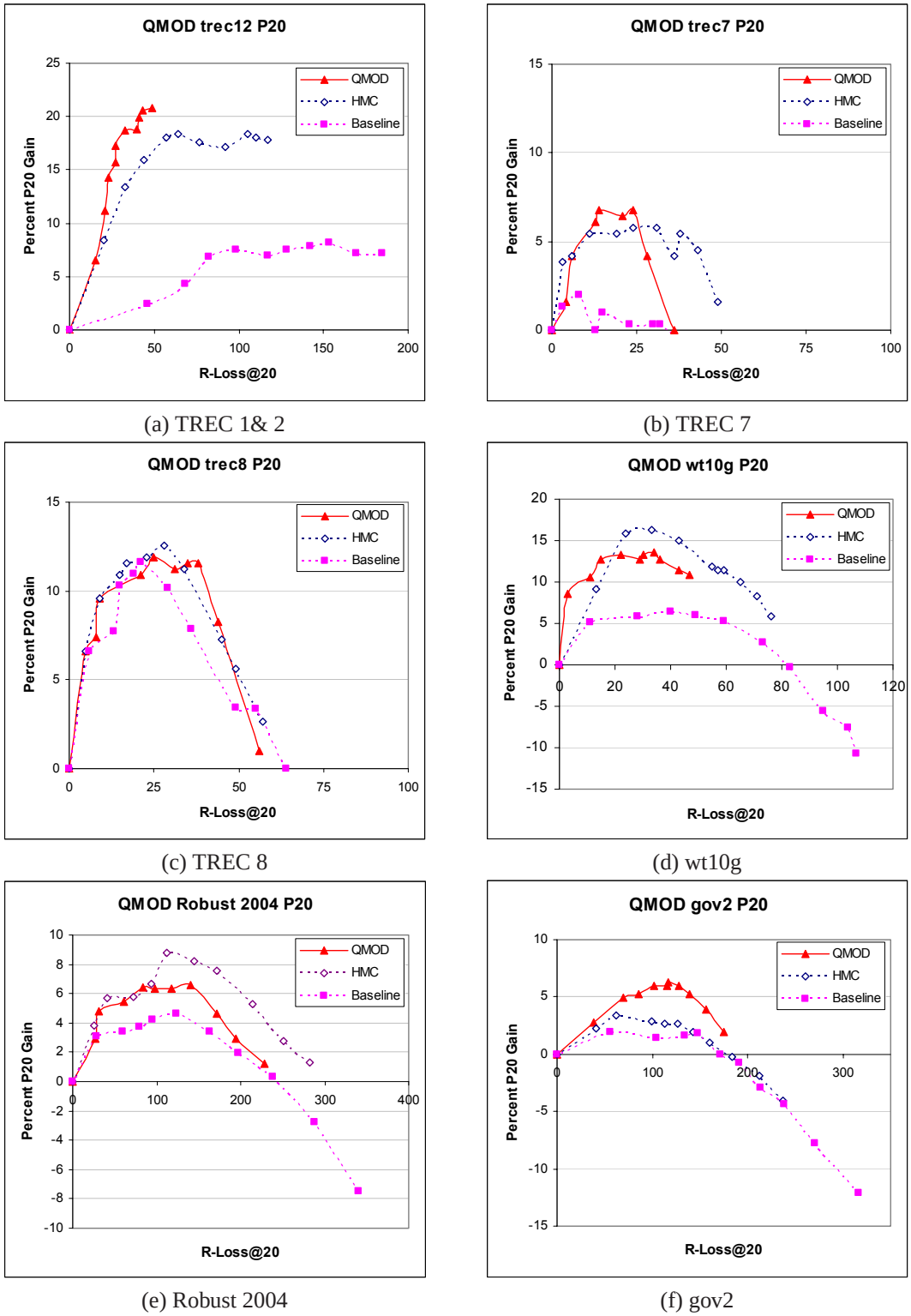


Figure 6.7: Risk-reward tradeoff curves for six TREC topic sets using P20 and R-Loss@20 (instead of MAP and R-Loss). The baseline feedback model is the Indri Relevance Model. Tradeoff curves that are *higher and to the left* give a better risk-reward tradeoff. Curves are plotted with points at α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.

algorithm. The dashed (pink) line is the curve given by the baseline expansion algorithm, which is the Indri Relevance model described in Section 3.4.2. Tradeoff curves that are *higher and to the left* give a better risk-reward tradeoff. For comparison with the QMOD model combination method, we have included HMC, the heuristic method described and evaluated in Chap. 3. The inputs to HMC and QMOD, query variants and resampled document sets, were the same.

It is evident that, except for one brief segment at the end of the Robust2004 curve, the QMOD tradeoff curve dominates the corresponding baseline curve for *any* value of the interpolation parameter α, on any topic set. In fact, most of the time the QMOD curve is significantly above and left of the baseline curve. This means that no matter what risk-reward tradeoff the baseline expansion model provides, the QMOD algorithm can always provide a better one, so that the same average precision is available, but with lower – and in most cases, significantly lower – downside risk. This also implies that the optimal MAP gain available with QMOD is always higher than the corresponding optimal baseline MAP performance.

The optimal MAP gain for the robust curve tends to occur at higher values of α – about one α-step of 0.1 – than the baseline method. (Recall that higher α values mean that less of the original query is used.) For example, we have the following optimal α values for QMOD vs baseline respectively: Robust2004: 0.7 vs 0.6; trec12: 0.8 vs 0.8; wt10g: 0.5 vs 0.3; trec8a: 0.7 vs 0.6; trec7a: 0.9 vs 0.8; and gov2: 0.4 vs 0.3. The average optimal $\alpha = 0.67$ for the robust QMOD model compared to $\alpha = 0.57$ for the baseline. In fact, if we choose a standard operational setting for QMOD of $\alpha = 0.6$, the result are statistically as good or better than the optimal α setting for the corresponding baseline run.

Even in the extreme case when the initial query model is discarded and only the feedback model is used ($\alpha = 1.0$) – we call this the curve's 'endpoint' – the performance of the QMOD algorithm is still reasonably good. Generally, the endpoints of the baseline algorithm give the worst performance – even if the MAP gain is high at an endpoint, the same MAP gain is available with much lower downside risk at a smaller value of α. The same is true for the QMOD tradeoff curve, but for every test set the endpoint of the QMOD algorithm is above and left of the baseline endpoint, providing higher relative reward with lower risk. In general, because the QMOD feedback model is more reliable, it is safer to choose higher operational values of α.

For an alternate view, curves using P20 and R-Loss@20 as reward and risk measures are

shown in Figure 6.7. It is notable that QMOD never hurts P20 for any α on any collection we tried. HMC (heuristic) only suffers a small loss on the gov2 collection for $\alpha = 1.0$. The Relevance model baseline hurts P20 significantly for the robust2004, wt10g and gov2 collections at around $\alpha = 0.7$ and above (depending on the collection).

In comparing the QMOD and HMC methods, both dominate the baseline feedback algorithm, but have slightly different risk-reward properties. Generally, QMOD gives lower-risk solutions. For example, as shown in Table 6.2 R-Loss@20 is consistently lower than HMC, with little loss in MAP. Compared to HMC tradeoff curves, QMOD curves tend to be shrunk slightly toward the origin, with endpoints ($\alpha = 1.0$) reflecting consistently lower R-Loss, especially when measured by P20. The penalty for QMOD is a minor reduction in maximum achievable MAP gain, which is about 2–3% higher with HMC. One likely explanation for this difference is that QMOD is a true selective expansion method: 15–20% of queries are simply not expanded because of high estimated risk. HMC does not have this 'hard' selection ability and always expands, resulting in a somewhat less conservative strategy.

General retrieval measures Table 6.2 compares average precision, R-Loss, and RI statistics for the initial, baseline, QMOD, and HMC (Chapter 3) feedback methods for specific choices of $\alpha = 0.5$ (the standard setting). For all six collections, at $\alpha = 0.5$ the average precision and P20 for QMOD are statistically equal or superior to the baseline expansion, while QMOD also reduces the number of relevant documents in the top 20 lost to failures (R-Loss@20) by amounts ranging from 34.5% (TREC 8) to 76.9% (TREC 1& 2).

Note that the no-expansion case provides the initial relevant documents, serving as the baseline for R-Loss, so R-Loss is always zero for the no-expansion case. The total number of relevant documents is shown in the denominator of the R-Loss fraction.

Comparing QMOD and HMC, the overall MAP gains over the original query performance are comparable: the difference in percentage gain is less than 5% on average, and differences in P20 are even smaller. However, QMOD is somewhat more robust, achieving lower R-Loss@20 scores than both Base-FB and HMC-FB on all collections.

Looking at the simple fraction of net queries helped using the Robustness Index (RI), QMOD at $\alpha = 0.5$ outperforms the baseline at $\alpha = 0.5$ on 5 out of 6 collections, and has equal performance for TREC 8. QMOD has higher RI than HMC on four out of six collections.

Collection		NoExp	Base-FB ($\alpha = 0.5$)	QMOD-FB ($\alpha = 0.5$)	HMC-FB ($\alpha = 0.5$)
TREC 1&2	MAP	0.1762	0.2317 (+31.9%) $^{\mathbf{N}}$	0.2346 (+33.2%) $^{\mathbf{N,E}}$	**0.2472** (+40.3%) $^{\mathbf{N,E}}$
	P20	0.4217	0.4483 (+6.94%) $^{\mathbf{N}}$	0.4945 (+17.3%) $^{\mathbf{N,E}}$	**0.4990** (+18.3%) $^{\mathbf{N,E}}$
	R-Loss@20	0/366	117/366	**27/366** (-76.9%)	64/366 (-45.2%)
	RI	0	0.4844	**0.5859**	0.5781
TREC 7	MAP	0.1830	0.2079 (+13.8%) $^{\mathbf{N}}$	0.2106 (+15.1%) $^{\mathbf{N}}$	**0.2165** (+18.3%) $^{\mathbf{N,E}}$
	P20	0.3456	0.3467 (+0.3%)	**0.3689** (+6.8%) $^{\mathbf{N,E}}$	0.3656 (+5.9%) $^{\mathbf{N,E}}$
	R-Loss@20	0/57	23/57	**12/57** (-47.8%)	24/57 (+4.5%)
	RI	0	0.4146	**0.5610**	0.4634
TREC 8	MAP	0.1920	0.2220 (+15.5%) $^{\mathbf{N}}$	0.2199 (+14.5%) $^{\mathbf{N}}$	**0.2288** (+19.2%) $^{\mathbf{N,E}}$
	P20	0.3213	0.3585 (+11.8%) $^{\mathbf{N}}$	**0.3660** (+13.9%) $^{\mathbf{N}}$	0.3596 (+11.9%) $^{\mathbf{N}}$
	R-Loss@20	0/76	29/76	**19/76** (-34.5%)	23/76 (-20.6%)
	RI	0	0.4286	0.4286	**0.4762**
wt10g	MAP	0.1747	0.1830 (+5.2%)	**0.1990** (+14.0%) $^{\mathbf{N,E}}$	0.1984 (+13.6%) $^{\mathbf{N,E}}$
	P20	0.2228	0.2340 (+5.4%)	**0.2512** (+12.7%) $^{\mathbf{N,E}}$	0.2494 (+11.9%) $^{\mathbf{N,E}}$
	R-Loss@20	0/158	59/158	**29/158** (-50.8%)	55/158 (-6.7%)
	RI	0	-0.0270	**0.2703**	0.1892
robust2004	MAP	0.2152	0.2441 (+13.5%) $^{\mathbf{N}}$	0.2451 (+13.9%) $^{\mathbf{N,E}}$	**0.2538** (+17.9%) $^{\mathbf{N,E}}$
	P20	0.3252	0.3397 (+4.5%) $^{\mathbf{N}}$	0.3458 (+6.3%) $^{\mathbf{N}}$	**0.3538** (+8.8%) $^{\mathbf{N,E}}$
	R-Loss@20	0/394	124/394	**98/394** (-21.0%)	112/394 (-9.7%)
	RI	0	0.3364	0.3773	**0.3818**
gov2 (2004–2006)	MAP	0.2736	0.2907 (+6.5%) $^{\mathbf{N}}$	**0.3004** (+9.8%) $^{\mathbf{N,E}}$	0.2959 (+8.1%) $^{\mathbf{N,E}}$
	P20	0.5214	0.5214 (+0.0%)	**0.5524** (+6.0%) $^{\mathbf{N,E}}$	0.5352 (+2.6%) $^{\mathbf{N,E}}$
	R-Loss@20	0/575	171/575	**116/575** (-32.2%)	126/575 (-26.3%)
	RI	0	0.0922	**0.2624**	0.1915

Table 6.2: Performance comparison of baseline (Base-FB) feedback, robust (QMOD-FB) feedback, and heuristic model combination (HMC-FB) feedback from Chapter 3. Precision improvement shown for Base-FB, QMOD-FB, and HMC-FB is relative to using no expansion. R-Loss change for QMOD and HMC are relative to baseline expansion (negative change is good). For Robustness Index (RI), higher is better. Significant differences at the 0.05 level using the Wilcoxon signed-rank test are marked by **N** and **E** superscripts, for improvement over NoExp and Base-FB respectively.

Robustness histograms Robustness histograms were introduced in Section 3.4.4 and provide a detailed look at how badly queries were hurt and helped by an expansion algorithm. Figure 6.8 gives the combined robustness histogram for $\alpha = 0.5$ for QMOD (dark) and the baseline (light). This makes clear that the worst failures – cases where a query's average precision was hurt by more than 60% – have been virtually eliminated by the QMOD algorithm, while the upside gain distribution remains very similar to the baseline gains. The gov2 corpus was most challenging, with four queries remaining with greater than 60% AP loss after expansion. The reasons for this require further analysis, but QMOD's overall gains in robustness were still significant (Table 6.2).

The most noticeable differences in gains are a reduction in the highest category (more than 100% AP gain) and an increase in the lowest gains (0 to 10%). Both of these are due to the selective expansion mechanism of the QMOD algorithm, in which queries that are deemed to risky to expand are left alone, resulting in a zero AP gain.

6.4.3 Parameter and constraint sensitivity

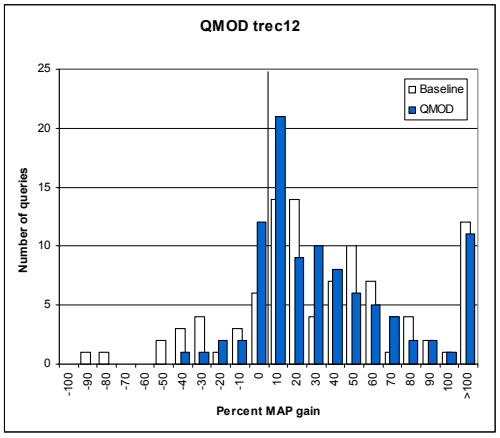
The QMOD convex program has a number of control parameters, which are summarized in Table 6.1 along with their default values. In this section we perform a sensitivity analysis for each of these parameters to see how they affect the risk-reward tradeoff as they are changed. In particular, we look at which constraints and parameters are most influential in finding robust solutions compared to the baseline expansion.

Bi-objective parameter (κ)

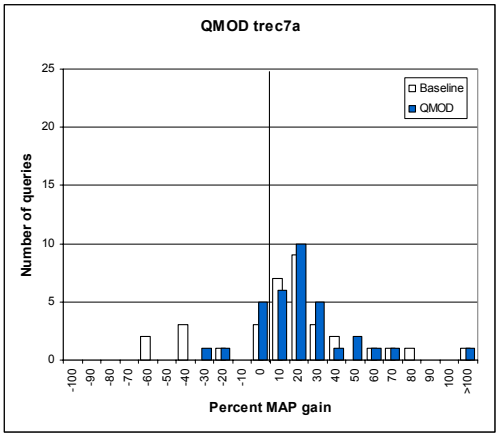
The bi-objective parameter κ dictates the relative weights given to the two objectives: finding terms with high initial relevance weights (node assignment costs in the graph) versus terms with low individual and conditional risk. Figure 6.9 shows a complete family of solutions as the parameter is increased from zero. Note how the proportion allocated to the original query terms *parkinson* and *disease* remains balanced for all solutions. The most effective range for κ is 0.5 to 1.25, which balances the original query terms with the expansion terms more or less equally. The default value we use for evaluation is $\kappa = 1.0$.

Query term support constraint

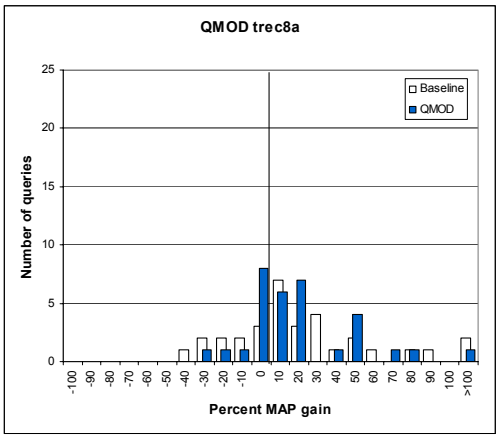
Varying the β parameter, which constrains the minimum allowable label value for the initial query terms, has a dramatic effect on both risk and reward, as Figure 6.10 shows. For all collections we tried, highly dominant tradeoff curves were obtained for β values close to



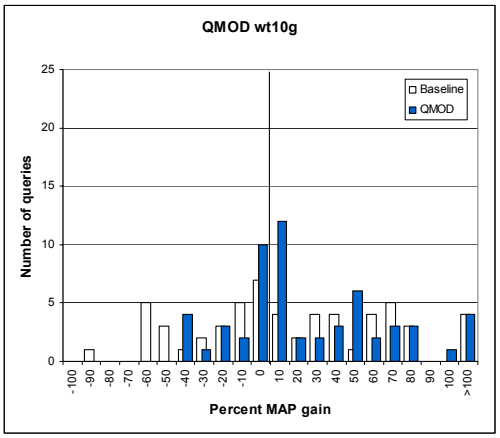
(a) TREC 1&2



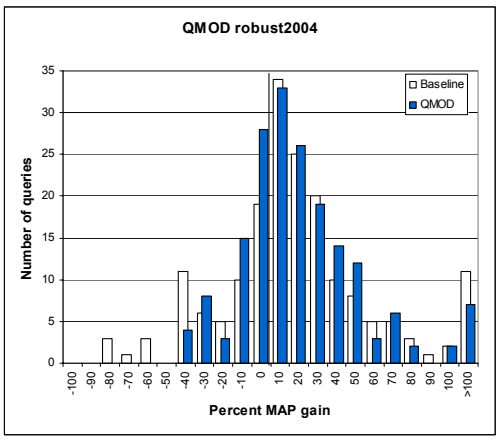
(b) TREC 7



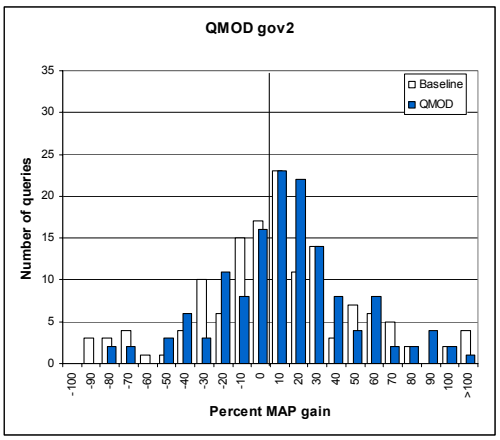
(c) TREC 8



(d) wt10g



(e) robust2004



(f) gov2

Figure 6.8: Comparison of expansion robustness for six TREC collections, showing how the robust QMOD version hurts significantly fewer queries, seen by the greatly reduced tail on the left half (queries hurt). (Recall that MAP performance of QMOD is also as good or better than baseline.) The histograms show counts of queries, binned by percent change in MAP. The dark bars show robust expansion performance using the QMOD convex program with default control parameters. The light bars show baseline expansion performance.

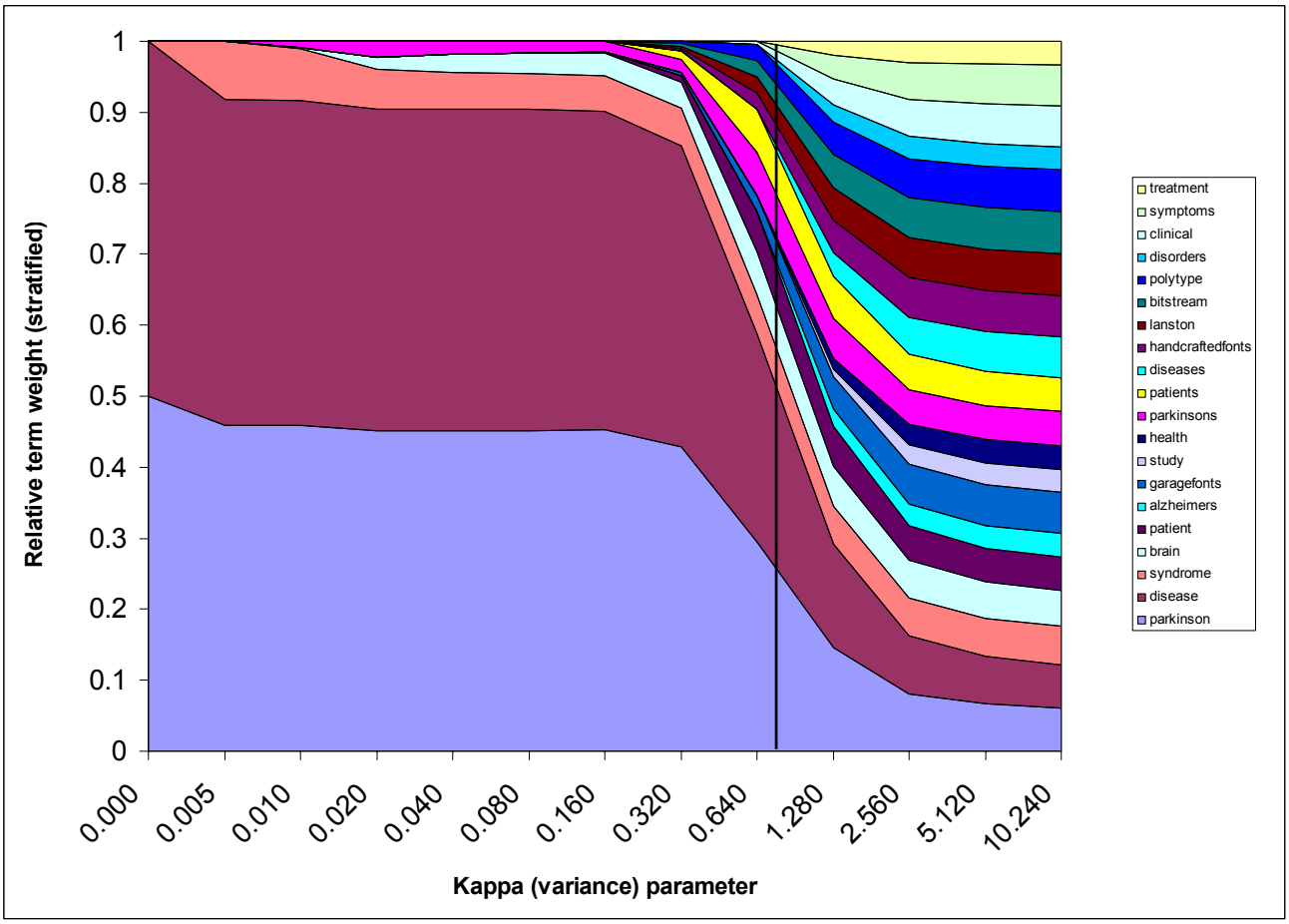
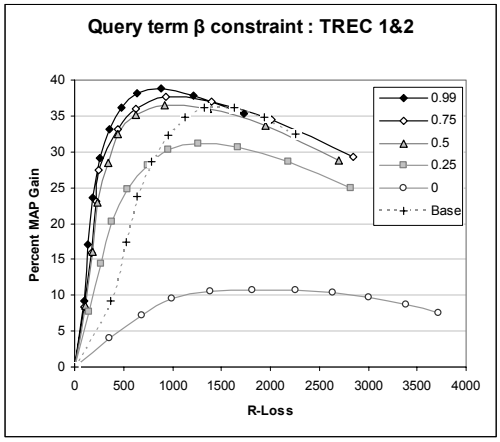
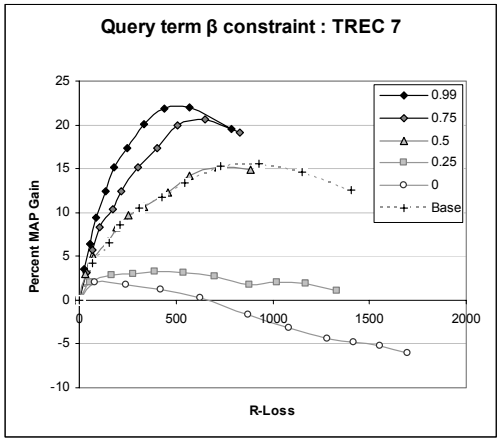


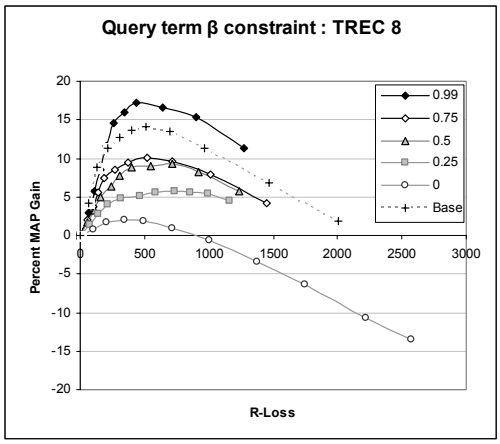
Figure 6.9: Example showing a family of solutions for a simple quadratic program as a function of the covariance objective weight κ (x-axis). When κ is close to zero, emphasis is on the relevance maximization objective (query terms and terms with highest relevance weights). As κ is increased, more weight is given to the risk (covariance) minimization objective. Each vertical 'slice' represents the output of the CVXOPT solver running the QMOD quadratic program for a particular value of κ, showing elements of the x solution vector (final relative term weights). The vertical line shows a typical default value of $\kappa = 0.75$. The query in this example is TREC topic 454: "parkinson's disease". (Some terms, such as 'garagefonts' and 'bitstream' are noise terms.)



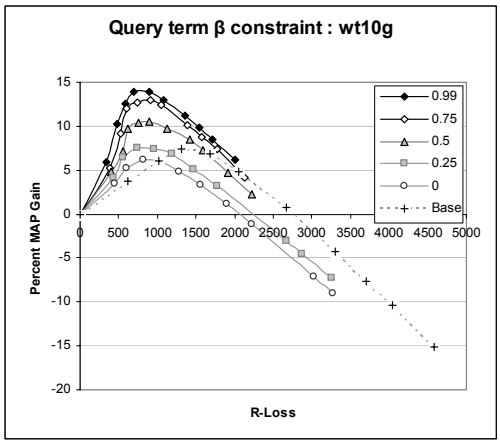
(a) TREC 1&2



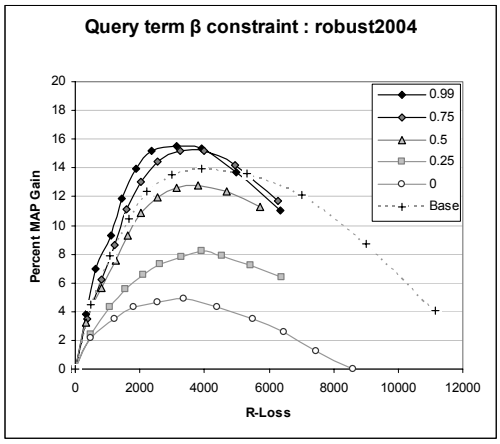
(b) TREC 7



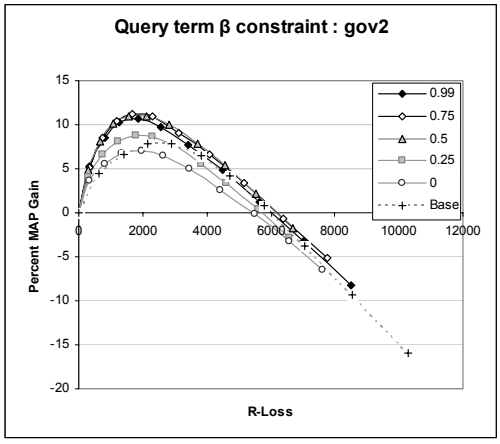
(c) TREC 8



(d) wt10g



(e) Robust 2004



(f) gov2

Figure 6.10: The effect on the risk-reward tradeoff curve of varying the query term weight constraint (β), with other QMOD parameters kept at default values. The baseline expansion tradeoff curve is also shown (dotted line).

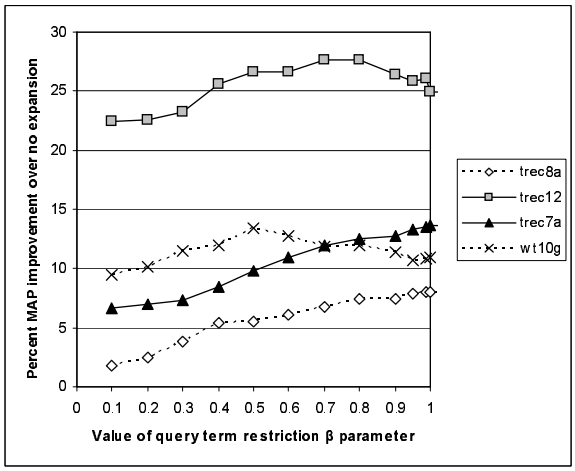
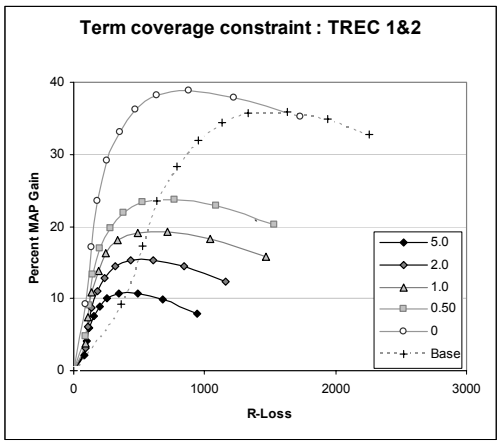


Figure 6.11: The effect on MAP gain – ignoring risk – of varying the restriction on minimum query term label value, represented by the parameter β. The baseline expansion tradeoff curve is also shown (dotted line).

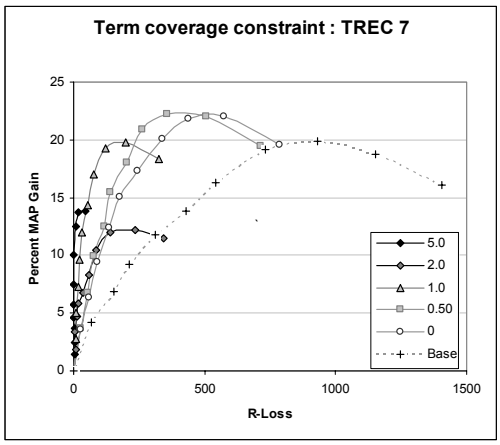
1.0, meaning that it is important for the QMOD solution to strongly predict the initial query terms. Specific per-collection optimal β values varied from 0.75 to 1.0 between collections, but the difference between optimal performance and performance close to 1.0 tended to be small. (We discuss this further below.) Choosing $\beta = 0.99$ was a consistently good choice for all collections. Interestingly, risk (in terms of R-Loss) changes little as β increases from 0.5 and above, even though reward (MAP gain) changes substantially. On the other hand, as β decreases below 0.5, risk increases and reward continues to decrease substantially. The β constraint by itself does not provide a complete solution: other parts of the convex program, such as the covariance parameter γ, act to reduce risk in tandem with the β constraint and we show this in the following sections.

We currently set the minimum label value for query term $\boldsymbol{q}_i$ to the same constraint level $\beta_i = \beta$ for all query terms. An interesting item for future work would be to examine methods for setting term-specific β_i values, so that, for example, rare terms may benefit from higher β_i constraints.

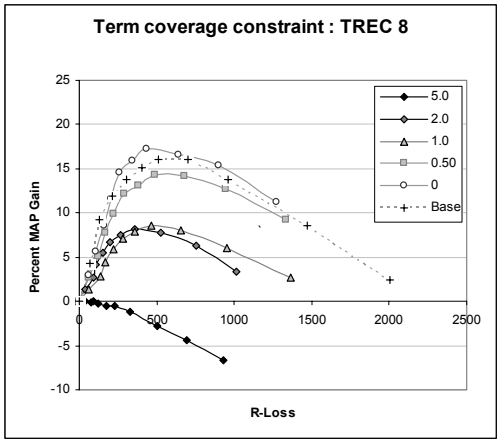
Figure 6.11 provides another view of β sensitivity that ignores the risk tradeoff, showing MAP gain only, as a function of β (with the feedback interpolation parameter $\alpha = 0.5$) for different collections.



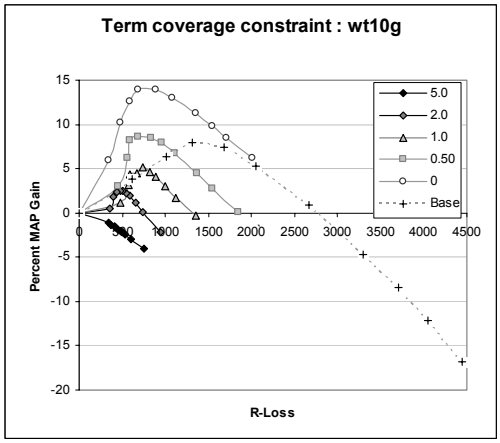
(a) TREC 1&2



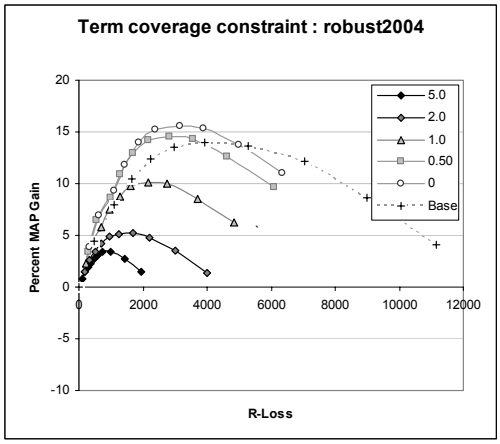
(b) TREC 7



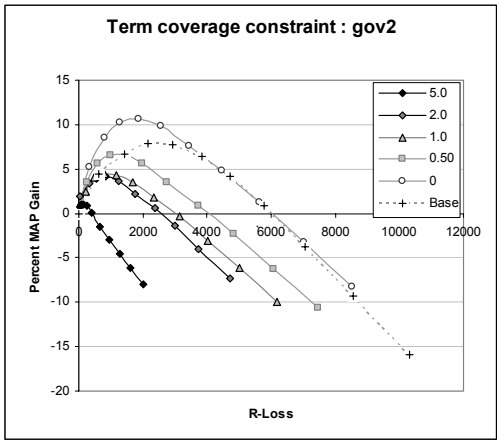
(c) TREC 8



(d) wt10g



(e) Robust 2004



(f) gov2

Figure 6.12: The effect on the risk-reward tradeoff curve of varying the term coverage constraint, represented by the parameter ζ_i. The baseline expansion tradeoff curve is also shown (dotted line).

Term coverage constraint

Figure 6.12 shows the effect of varying the term coverage parameter ζ_i. Higher values of ζ_i act to strengthen the constraint: the feasible set is restricted to only those solutions having multiple good expansion terms near to every term in the query, giving much more risk-averse behavior of the program. Indeed, we can see that risk does usually shrink significantly as ζ_i is increased. However, for this parameter there is a corresponding decrease in MAP gain due to the increased number of infeasible queries (zero AP gain) as the program becomes more conservative. In practice, a ζ_i value between zero and 0.5 allows a beneficial reduction in risk. Our default setting is to omit the constraint, setting $\zeta_i = 0$. Values of ζ much greater than 1.0 are likely to be too conservative for most scenarios. An interesting item for future work would be to experiment with feasible sets that penalize term coverage in different ways instead of rewarding it.

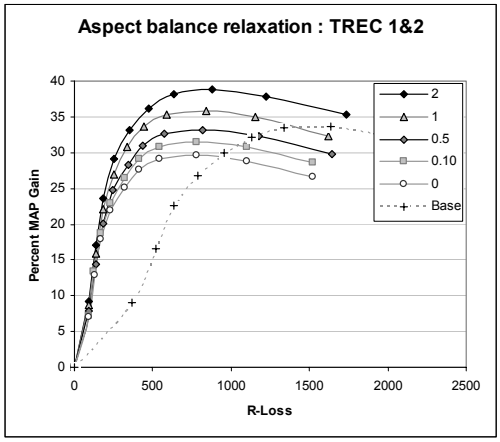
Aspect balance tolerance

Figure 6.13 shows the effect of relaxing the aspect balance tolerance parameter ζ_μ upward from zero. The general result of enforcing a strong centering constraint ($\zeta_\mu = 0$) is to shrink the curve downward, but with little change in risk. This result occurs because as the constraint becomes stronger, the QMOD program becomes more conservative and more reluctant to touch the initial query. As a result, overall MAP gain shrinks, but the quality of the remaining balanced expansions is still high.

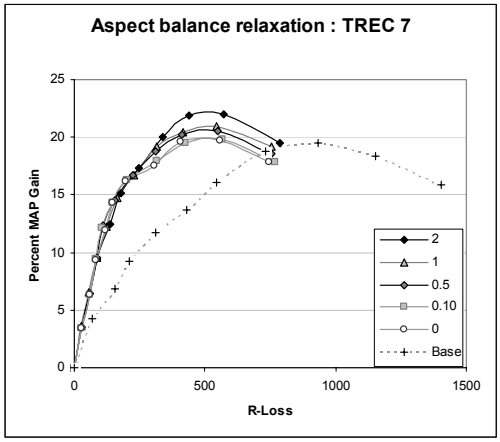
We found that values of ζ_μ around 2 give consistently good results for all collections. Increasing ζ_μ beyond 2 yields little further improvement.

Covariance parameter γ

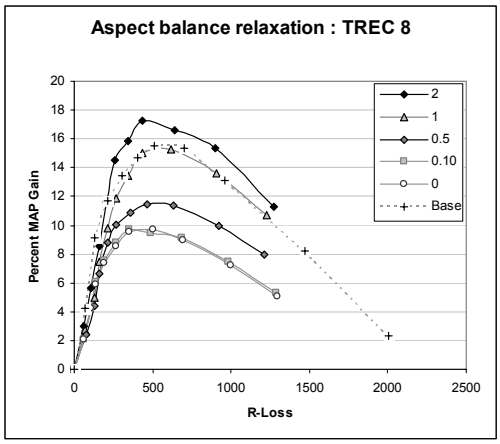
Recall that we decomposed the covariance matrix Σ as $\Sigma = D + \gamma \cdot E$ where D is a diagonal matrix of individual term risk factors, and E a symmetric matrix with e_{ij} being the conditional risk of using term i given we have already selected term j. The parameter γ controls the relative influence of the off-diagonal elements of Σ compared to individual term risk. Our hypothesis is that including the conditional term risk in our QMOD program improves the solution, and thus that there is some optimal value or range of γ is greater than zero. Figure 6.14 shows the effect of varying γ across all six collections. We observe that reducing γ does reduce MAP gain, but simultaneously gives a significant reduction of risk (as measured by R-loss); the slope of the tradeoff curve, for the main operational range from 0



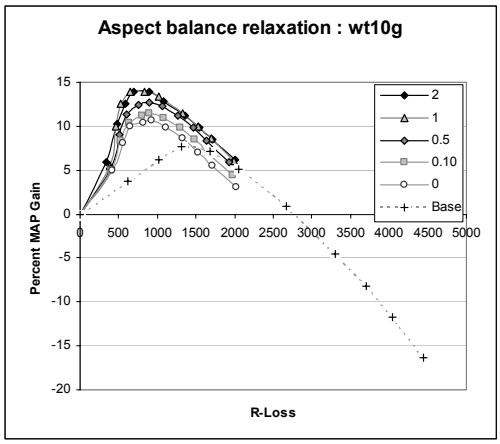
(a) TREC 1&2



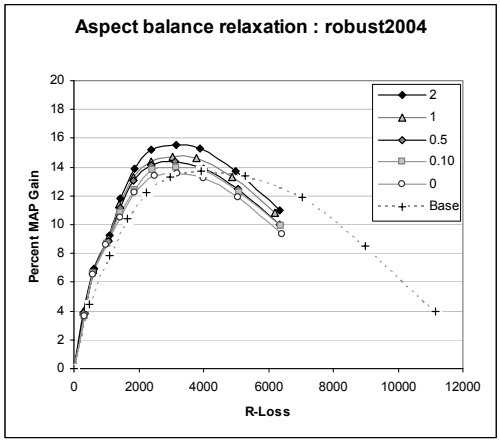
(b) TREC 7



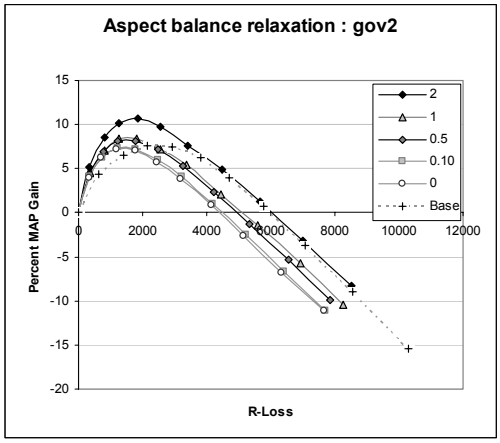
(c) TREC 8



(d) wt10g

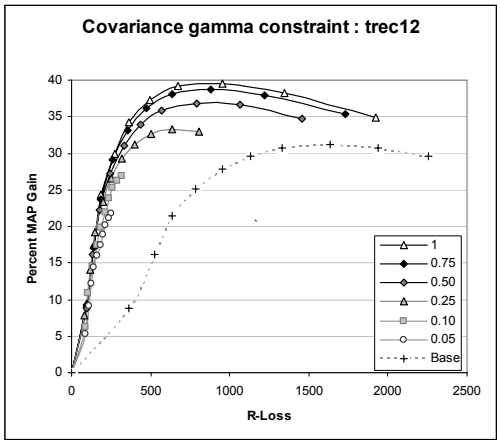


(e) Robust 2004

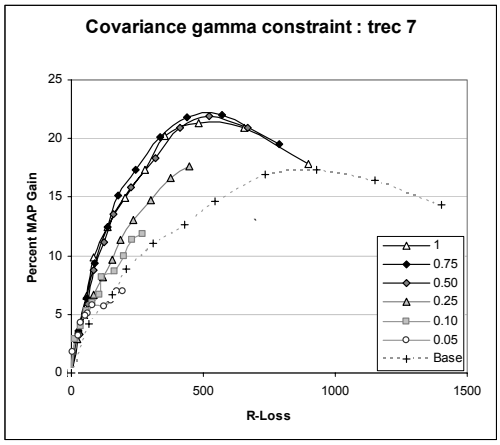


(f) gov2

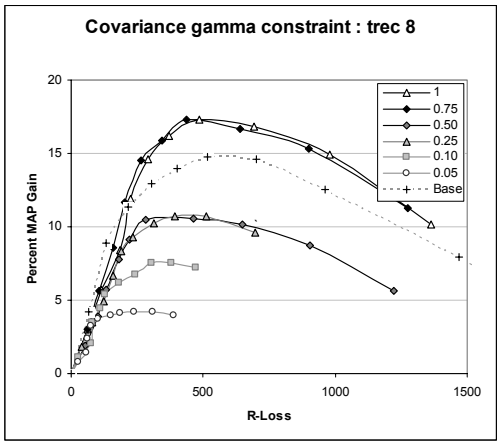
Figure 6.13: The effect on the risk-reward tradeoff curve of relaxing the aspect balance condition by increasing ζ_μ from 0 to 2, with other QMOD parameters kept at default values. ($\zeta_\mu = 0$ forces exact centering.) The baseline expansion tradeoff curve is also shown (dotted line).



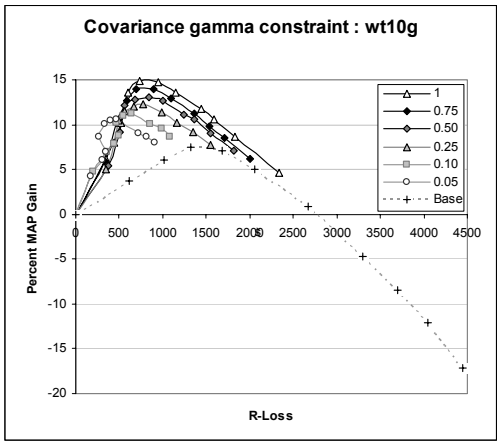
(a) TREC 1& 2



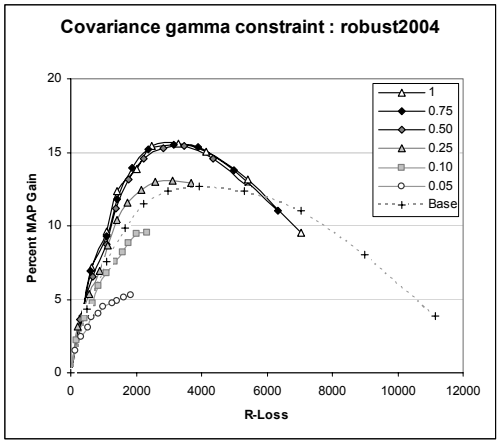
(b) TREC 7



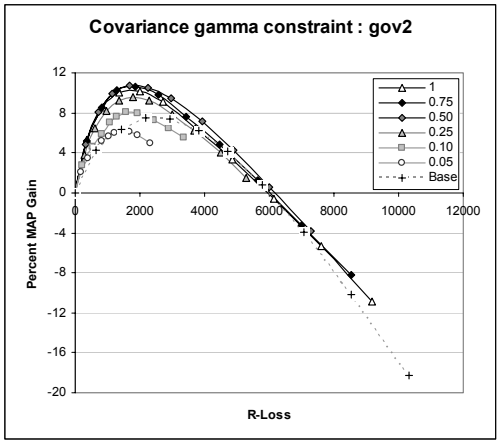
(c) TREC 8



(d) wt10g



(e) robust2004



(f) gov2

Figure 6.14: The effect of the covariance parameter (γ) on the risk-reward trade-off curve. As γ is increased, the off-diagonal elements of the covariance matrix, representing term dependencies, are given more weight. The baseline expansion tradeoff curve is also shown (dotted line).

to 0.5, remains roughly the same. Thus, values of γ close to zero result in robust but very conservative solutions. As γ increases toward 1.0, the tradeoff curve expands upward and to the right, until between 0.75 and 1.0 it reaches an approximate maximum MAP gain. Values of γ much beyond 1.0 tend to preserve this maximum gain and curve shape, but pull and extrapolate the curve right, extending the risk while providing negligible additional little MAP gain. We find that a value of $\gamma = 0.75$ is a good default value for all evaluated collections.

6.4.4 Effect with an alternate expansion algorithm

We also applied QMOD to the results of a different strong expansion algorithm. We replaced the baseline Indri method (Relevance model) with a Rocchio-style vector space method in which the top *k* document vectors were given equal weight and used a *tf.idf* representation. The same query variants and document resampling were used as in the Indri experiments. The tradeoff curves are shown in Figure 6.15.

The Rocchio baseline had slightly stronger performance than the Relevance model baseline on the trec12, trec7, and trec8 collections. QMOD still achieved a gain on the initial half of the trec12 and trec7 curves. QMOD continued its strong performance on the two Web collections. As it did with the Relevance model baseline, QMOD dominates the Rocchio *tf.idf* baseline for the wt10g and gov2 collections.

6.4.5 Tolerance to poor baseline expansion algorithm

As we did in Section 3.4.6 with the RS-FB with heuristic model combination, we tested QMOD's tolerance for noise by applying it to the same very poor baseline algorithm, namely, a Rocchio scheme that ignores term frequency (tf) and uses only idf in the term representation. This results in a very noisy expansion model dominated by rare terms that are poor discriminators for relevance. The results for two representative collections, TREC 7 and wt10g, are shown in Figure 6.16. For comparison, the results for RS-FB (Chapter 3) are also shown. As we saw in Section 3.4.6, this idf baseline is very weak: MAP loss at $\alpha = 1.0$ was worse than -80% in all cases. However, applying QMOD successfully limited the damage caused by the weak expansion baseline, in a manner much more effective than RS-FB. At $\alpha = 0.5$, for TREC 7a, MAP loss is reduced from -11.8% to almost zero (0.88%) with reduction in R-Loss from 1136 to 390. For wt10g, MAP loss is reduced from -35.1% to -6.1% with reduction in R-Loss from 5485 to 1703. We have omitted the other

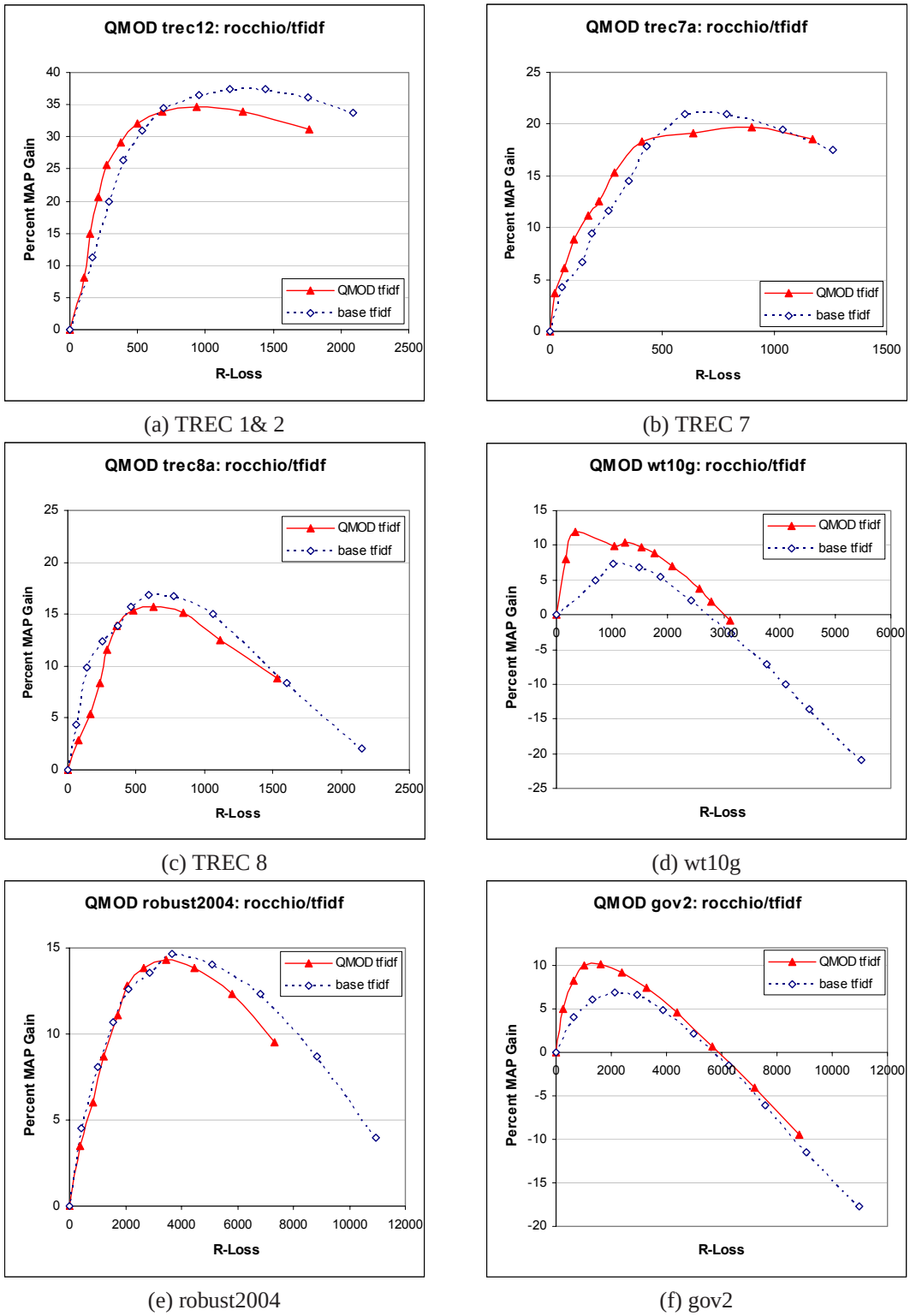
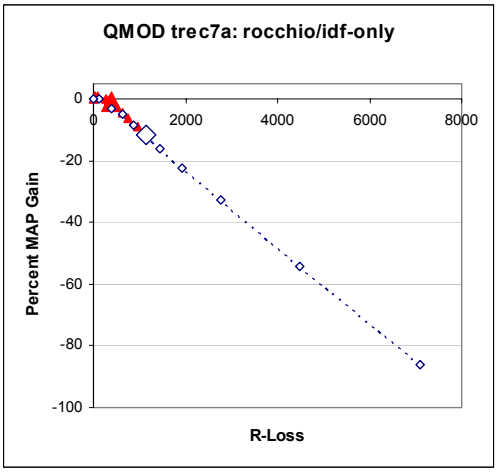
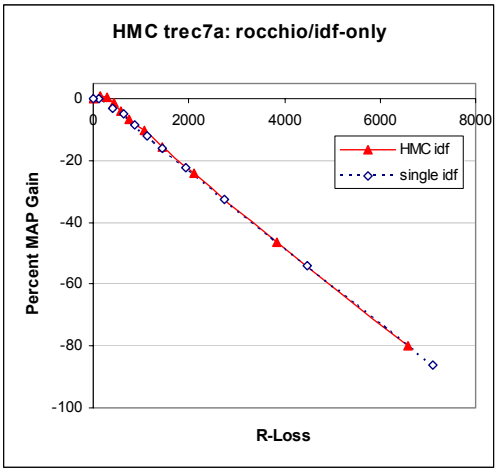


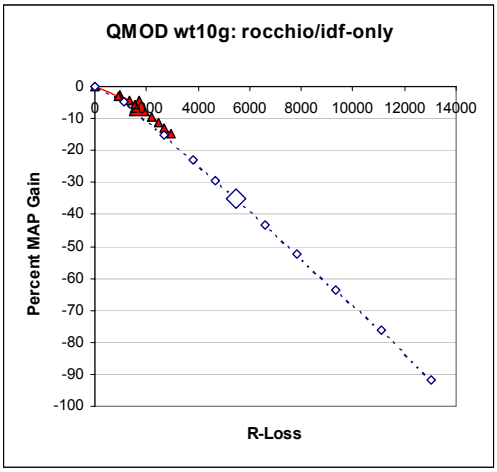
Figure 6.15: The effect on risk-reward tradeoff curves of applying QMOD (solid line) to a Rocchio-style expansion algorithm (dotted line) instead of the default Relevance model baseline. Tradeoff curves that are *higher and to the left* are better. Points are plotted in α-increments of 0.1, starting with $\alpha = 0$ at the origin and increasing to $\alpha = 1.0$.



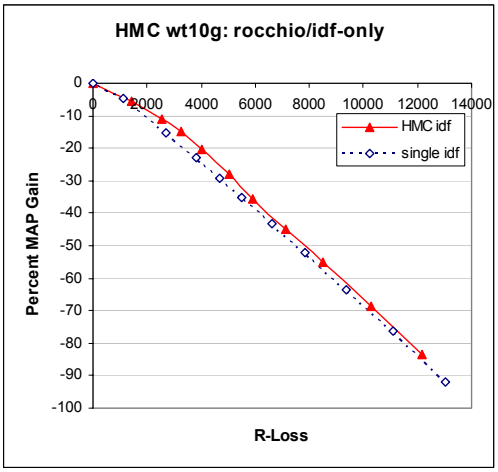
(a) TREC 7 (QMOD)



(b) TREC 7 (RS-FB)



(c) wt10g (QMOD)



(d) wt10g (RS-FB)

Figure 6.16: Risk-reward tradeoff curves for two representative TREC topic sets, showing the much greater tolerance of the convex QMOD algorithm (left) to noise from a poor baseline expansion algorithm. For comparison, the weak tolerance of RS-FB (Chap. 3) for the same *idf* baseline is shown (right). The point corresponding to $\alpha = 0.5$ for each method is enlarged for visibility. Results for other collections are similar.

four standard collections because their results are similar: the MAP loss from QMOD at $\alpha = 0.5$ is typically between 0% and -5%, versus a baseline MAP loss on the order of -20% to -40%.

While this noise scenario is deliberately chosen to be extreme, it serves to clearly illustrate the greatly increased noise tolerance that QMOD achieves due to its selective nature, compared to the non-selective RS-FB.

6.4.6 Calibration of feasible set

If the constraints of a convex program are well-designed for stable query expansion, the odds of an infeasible solution should be much greater than 50% for queries that are risky. In those cases, the algorithm will not attempt to enhance the query. Conversely, the odds of finding a feasible query model should ideally increase for thoese queries that are more amenable to expansion.

Across all collections, 17% of the queries had infeasible programs [9]. We binned these infeasible queries according to the actual gain or loss that would have been achieved with the baseline expansion, normalized by the original number of queries appearing in each bin when the (non-selective) baseline expansion is used. This gives the log-odds of reverting to the original query for any given gain/loss level.

The results are shown in in Figure 6.17. As predicted, the QMOD algorithm is more likely to decide infeasibility for the high-risk zones at the extreme ends of the scale. Furthermore, the odds of finding a feasible solution do indeed increase directly with the actual benefits of using expansion, up to a point where we reach an average precision gain of 75% and higher. Beyond that point, such high-reward queries are also considered high risk by the algorithm, and the likelihood of reverting to the original query increases dramatically again. In previous work, ([Carpineto et al. 2001a], p. 18) found that queries with high initial precision could hardly be improved upon, and suggested that ‘selective policies for query expansion... (should) focus on queries that are neither too difficult nor too easy.’. Our analysis makes clear that the feasible set and thus the selective expansion behavior of the convex algorithm is well-calibrated to the true expansion benefit.

While this is a strong result, we are still losing some large gains for some queries. Given that these queries have a particularly extreme (positive) response to query expansion, identifying them is likely to be an easier task than trying to predicting expansion success

[9] We used a maximum of 100 convergence steps.

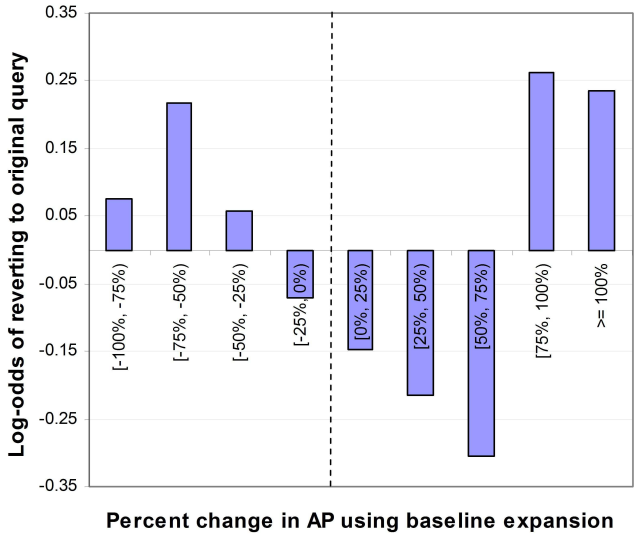


Figure 6.17: The log-odds of reverting to the original query as a result of selective expansion. Queries are binned by the percent change in average precision if baseline expansion were used. Columns above the line indicate greater-than-even odds that we revert to the original query.

for any possible query. Features that distinguish extreme expansion success may be similar to the kinds of features explored for query difficulty, such as the quality and stability of the initial results clustering, expansion term clustering, the specificity of the query terms, and so on. These features may then be incorporated as constraints in the QMOD program which are adjusted for each query.

6.5 Discussion

The robust QMOD estimation algorithm obtained consistently better tradeoff curves than the baseline expansion algorithm for the collections we evaluated. In most cases, the gains are quite striking: for example, on the 150 TREC 1&2 topics the QMOD algorithm achieves higher MAP gain at $\alpha = 0.6$ than the optimal baseline α, while losing less than half the number of relevant documents due to expansion failure. QMOD also outperformed the baseline according to the percentage gain in queries helped using the Robustness Index (RI). Overall, our results on six diverse test sets show that the QMOD solution dramatically reduces the downside risk of the baseline algorithm, without sacrificing its strong retrieval performance. We now analyze the factors contributing to these improved tradeoff curves,

and the implications for the design of future query expansion algorithms.

6.5.1 Factors in improved risk-reward tradeoffs

The best tradeoff curves were obtained using an intermediate mix of both objectives, and with all constraints active. Some parameters had a more dramatic effect on the tradeoff curve than others. The query term coverage parameter β was a highly influential constraint: it has a strong effect on MAP gain, but little effect on risk. Conversely, the conditional covariance parameter γ had a larger effect on risk reduction (and a weaker effect on MAP). Activating both of these together resulted in most of the improvement in the QMOD tradeoff curve. Other constraints such as the term centering constraint ζ_μ were less critical but acted to further shrink the risk of the tradeoff curve with little reduction in MAP. The term coverage constraint ζ_i also acted to increase the conservativism of the solution. In practice, a small value such as $\zeta_i = 0.1$ or less provides a good balance.

We found that good default QMOD parameter values were: highly constrained query term weights ($\beta = 0.99$), moderately relaxed term centering constraint ($\zeta_\mu = 2.0$), minimal term coverage constraint ($\zeta_i = 0.1$), intermediate use of conditional term risk ($\gamma = 0.75$) and roughly equal objective weighting ($\kappa = 1.0$). The value $\alpha = 0.4$ gives a safe tradeoff with smaller risk: the resulting combination comes within 10% or less of the optimal MAP for both algorithms.

6.5.2 Implications for query expansion

Our findings have the following implications for the design and analysis of future query expansion algorithms.

Query-anchoring of the expansion model. Our analysis shows that the best risk-reward tradeoff curves were obtained by expansion models in which the original query terms were highly weighted. Thus, strong anchoring to the original query appears to be a necessary, although not sufficient, condition for robust expansion models[10]. A key point here is that we are focused only on the support that exists for the query itself, and that this support need not be at the expense of the expansion terms: it is acceptable for many other terms to also obtain high label weights. Indeed, because we treat term weights as labels and not as probabilities, high weights on the initial query terms need not imply reduced weight on

[10]To simplify analysis, we assume a 'clean' initial query, i.e. that any typographic errors, misspellings, etc. in the original query have been corrected at an earlier stage of processing.

other terms.

This query-anchor finding is in accord with recent related work by Winaver et al. [2007], who reported that an extensive automatic search for the feedback model $\theta_{M^\star}$ that minimized the KL-divergence $KL(\theta_{M_i}||\theta_Q)$ to the initial query Q gave performance close to the manually optimized feedback model. Here, θ_Q is an unsmoothed language model constructed from the initial query Q and θ_{M_i} is a Jelinek-Mercer smoothed language model, constructed by interpolating the set of document language models θ_{D_i} of the 100 top-ranked documents using expanded query model M_i.

Selective expansion. Because of the uncertainty inherent in query expansion, we believe that a second necessary condition to obtain any robust query expansion algorithm is the use, in some form, of three basic steps that involve selective behavior at each stage.

1. The uncertainty in the 'correct' expansion model parameters should be captured by a set of multiple plausible model hypotheses, resulting in a 'bouquet' of alternatives. In our work, this is accomplished by using query variants to estimate different feedback models. In Chapter 3 we showed that when the number of alternate hypotheses is reduced, the robustness of the query expansion algorithm suffers.

2. A 'hard' selection process should eliminate implausible models completely. This role is performed in our work by the constraints defining a feasible set to a convex optimization problem. If necessary, all hypotheses except the observed query are sometimes rejected. The term centering and query term coverage constraints are examples of a weak and a strong constraint respectively.

3. A final 'soft' selection process is used to perform model combination on the remaining good models. Rather than assigning a single weight to all the terms in a model, we allow the algorithm to calculate weights that are term-dependent. This greatly increases the richness of the hypothesis space of possible solutions, making it more likely a good solution will be found. Although in theory this also greatly increases the number of model weights to be estimated, in practice the computational cost can be limited: for example, the vocabulary size can be limited using, say, an initial threshold on the top-*n* relevance weighted baseline terms. Model combination in this chapter is a natural result of solving the metric labeling objective: furthermore, expansion models with many good terms naturally result in lower relative weights to the original

query terms, while some queries have few or no good expansion terms. Effectively, this dynamically chooses the number of top-k final expansion terms (including zero terms), rather than forcing us to choose a fixed k in advance: this flexibility would be impossible with a simpler model-assigned weighting scheme.

Risk-reward curves. We regard the risk-reward tradeoff curve as an important new diagnostic tool for analyzing and comparing the effectiveness of expansion algorithms. Ideally, such curves should become standard in the evaluation of any query expansion or reformulation algorithm.

6.5.3 Comparing sampling and optimization approaches

The two main approaches we have introduced for improving robustness are the sampling-based approach of Chapter 3 and the convex optimization approach of this chapter. These methods are complementary, but quite different. While sampling acts to average over uncertainty, the QMOD framework can be used without probabilistic models and makes use of fixed estimates for objective and constraint parameters. We currently connect sampling and optimization by using perturbation kernels (obtained from sampling) as the covariance estimate in the optimization, but such a connection is not strictly required.

A key advantage of sampling-based methods is that they are a very general way to get at the sensitivity and uncertainty of virtually any observable result or parameter calculated by a retrieval algorithm. In Chapter 3 we showed that such sensitivity information was useful for improving the quality of expansion models. Moreover, we have a powerful set of existing statistical tools that can be applied to fit probabilistic models to the sampled information. These models in turn fit naturally with existing probabilistic retrieval frameworks. One disadvantage is the increased computational cost of obtaining the sampled results, although in Chapter 3 we discussed ways this expense could be mitigated.

At the start of this chapter, we already stated the many benefits of using an optimization framework for finding good expansion models. A limitation of our current approach, however, is that it does not yet fully incorporate the uncertainty information that could be available from sampling. This is partly by design, in order to start by exploring simpler models first. Ideally, however, we could go further to combine the strengths of sampling and optimization approaches by using a more general robust optimization framework that accounts for uncertainty in parameters (such as the aspect matrix A). Then, sampling methods could provide the estimates to quantify this uncertainty. Depending on our robust

optimization approach, these estimates could be in the form of distribution parameters or (distribution-free) moments obtained from sampled estimates for the uncertain variables, or could simply be finite sets of sampled parameter values.

6.6 Related Work

Historically, the problem of optimization under uncertainty has been the domain of operations research and economics. More recently, many other fields, including branches of computer science such as image processing (e.g. Tsuda & Ratsch [2005]) have made substantial algorithmic contributions. The common thread among these various fields is that of finding an optimal action or selection when we can only observe an imperfect signal describing the state of the world. This forces us to quantify uncertainty – typically by modeling it with a probability distribution – and to formulate decision rules about what optimality means when the data, or even the goal itself, are uncertain, and how to trade off benefits against risks.

One of the most prolific areas of research has been the field of computational finance, from which we have borrowed the risk/reward paradigm. The classic finance optimization problem is *portfolio allocation under uncertainty*. Initially pioneered by Markowitz [1952], the goal of portfolio optimization is to allocate a given budget over a set of securities in a way that not only maximizes the expected return of our investment, but also diversifies the portfolio to reduce risk. For example, we typically avoid buying too many highly correlated stocks in the same industry sector. The expected utility of the securities, and their covariance over times, is estimated from historical data, and the optimal portfolio is found as the solution to a quadratic optimization problem. This mean-variance optimization model has since been generalized and refined in numerous ways.

While the problems faced by portfolio managers and search engines both involve optimization under uncertainty, the finance scenario also has significant differences from information retrieval. First, although we would like to treat the query in some sense as a risk-free asset, we currently have no reliable way to quantify what the corresponding "rate of return" might be, which would require a reliable estimate of query difficulty. Second, we typically do not have extensive historical data to model variance. Instead, we must generate our own pseudo-training data for every query, using methods like the query variant strategy of Chap. 3. The extensive query logs generated from millions of user queries to Web search engines may prove to help in this regard.

6.7 Conclusions

Our work in this chapter extends the ideas and expansion framework introduced in Chapter 3 by adapting convex optimization methods for model combination. We show further improvements over the earlier heuristic approach in the quality of the final combined model based on the results of multiple query variants. Our approach is to cast the problem of query model estimation in terms of constrained metric labeling. By integrating relevance and risk objectives with additional set-based constraints to selectively reduce expansion for the most risky queries, our approach produces dramatic reductions in the downside risk of a strong baseline algorithm while retaining or exceeding its gains in average precision.

We introduced simple refinements to the basic metric labeling problem that capture constraints on the nature of query terms selected for an expansion. The quality of *aspect balance* restricts label weights so that all aspects (terms) of a query are covered more or less equally. The *aspect coverage* constraint specifies roughly how many nearby related terms, for any query term, must exist in the solution: allowing more expansion terms increases potential recall, but also potential expansion risk. The *term centrality* constraint prefers terms whose distances (say, as measured using the data perturbation kernel) to all of the query terms have low variance, and thus are more centrally located in kernel space. We also showed how other heuristic constraints, such as budget constraints also fit easily into this framework.

Because of the generality of our framework, a number of extensions and refinements to the basic program can be studied. For example, additional budget constraints may be added to constrain the total number of non-zero term weights in the solution, similar to methods from portfolio optimization [Lobo et al. 2007]. Second, sensitivity analysis of the constraints is likely to provide useful information for active learning: interesting extensions to semi-supervised learning are possible to incorporate additional observations such as relevance feedback from the user. Finally, there are a few additional parameters, such as kernel rescaling coefficients, and it would be interesting to determine the optimal settings. The values we use have not been extensively tuned, so that further performance gains may be possible.

In the concluding chapter that follows, we look beyond the domain of term weights to describe how our constrained optimization approach may be generalized to help solve difficult information retrieval problems in other domains.

Chapter 7

Conclusion

In essence, all work in this thesis flows from the logical implications of two key assumptions. First, that important entities such as queries and top-ranked document sets be treated as noisy observations of a latent random variable. Second, that the resulting posterior distributions – and expectations, covariance matrices, and other quantities derived from them – represent information about the critical but neglected dimension of *risk* which can be quantified and exploited to improve the robustness and precision of information retrieval algorithms in a very general way. As a concrete application of this approach, we make significant progress on a long-standing problem: improving the reliability of query expansion algorithms without reducing their overall effectiveness, while making few assumptions about the base expansion technique.

In practical terms, we implement this vision in two phases: first by using sampling – producing small numbers of query and document variants – to estimate feedback model risk and generate multiple feedback model hypotheses; and second, using an optimization framework that prunes and combines these model hypotheses to produce a robust, effective final expansion model.

Starting with the basic generalized retrieval framework described in Chapter 3, we achieved significant improvements in both precision and robustness over six standard TREC test collections. The best results in that chapter were obtained by combining leave-one-out query variation, bootstrap sampling over documents, and a heuristic model combination step derived from sample weighting heuristics in Monte Carlo integration. We then developed further improvements to model combination in Chapter 6 using a more principled, transparent and extensible framework based on convex optimization. This resulted in the

QMOD algorithm, which, unlike the heuristic model combination method, was much more tolerant of a poor baseline algorithm and could operate selectively to avoid risky expansion situations. We showed that, within the optimization step itself, the best results were obtained by combining several active constraints on the set of expansion terms, with individual and conditional term risk estimates. We obtained further incremental improvements by re-using term score variance information in Chapter 4 to derive perturbation kernels, a new type of similarity measure.

7.1 Significance of this work to the field of information retrieval

We believe this dissertation introduces important changes to the way people will view, implement and evaluate query expansion. First, query expansion methods no longer need to restrict themselves to greedy, threshold-based heuristics that neglect properties of the set of terms as a whole, such as aspect balance. Instead, we are free to think of expansion methods in the most natural way, namely, in terms of principled *set-based* criteria, and balanced trade offs between multiple competing objectives such as relevance and risk.

Another shift is our emphasis on quantifying the *risk* of query expansion instead of merely maximizing reward measures like mean average precision. For evaluation, we introduce estimation and analysis of risk and reward, focusing on downside risk, not just average-case performance. We include new evaluation methods such as risk-reward tradeoff curves, which we believe should become a standard method for analyzing and comparing expansion algorithms in the future.

To implement these ideas, we add two powerful new methods: the use of resampling for model estimation, and optimization with constraints for selective expansion. There is great freedom in particular in the relevance and risk objectives and metric constraints that may be used with the optimization framework of Chapter 6. Extensions and refinements to both resampling and optimization represent new research directions for IR in their own right[1]. In general, the principled, extensible theoretical framework we introduce is fruitful ground for future exploration of factors that interact and affect query expansion.

Our algorithmic contributions focus on *general-purpose* methods that can take existing

[1] For example, [Lee et al. 2008] very recently published a cluster-based refinement to document resampling that improves on the basic methods in [Collins-Thompson & Callan 2007]for pseudo-relevance feedback.

expansion algorithms and make them better. Most proposed improvements to query expansion only apply to a particular retrieval model. Our algorithms, on the other hand, treat the retrieval model or ranking function as a black box, which could be implemented using vector space models, inference networks, statistical language modeling, or other approaches. Thus, the techniques we introduce are broadly applicable.

We now give a more detailed overview of contributions than our original summary in Chapter 1.

7.2 Dissertation Summary

In the first part of this thesis, we discussed the issue of robustness in query expansion algorithms, and how it is important to distinguish average-case performance from worst-case performance when evaluating retrieval algorithms such as relevance feedback. Current feedback methods are unstable and while performing well on average, can still hurt many individual queries. We summarized the main causes of query drift, which is a key problem of existing expansion methods. These problems include poor initial retrieval, aspect imbalance, and unstable term quality. We discussed how these problems can be addressed with novel applications of sampling and convex optimization that can measure and account for *risk* as well as *reward* in searching for effective and robust query models.

7.2.1 Sampling methods

In order to incorporate risk as a factor in information retrieval algorithms, we introduced methods for estimating risk efficiently. Given some quantity, such as score variance, as a proxy for risk, we showed how the use of *sampling* in small amounts can produce useful estimates of variance.

In Chapter 2 we introduced another application of sampling, namely, calculating Monte Carlo estimates of important integrals that arise in document scoring. We derived document scoring in the Relevance Model as a form of importance sampling . Monte-Carlo-type estimates recur in the thesis for a number of useful integrals: the GRM formula for document scoring, document-smoothing kernels, and canonical similarity measures (data perturbation kernels).

We address the issue of noise terms in query expansion models by applying a generalized form of bagging to the top-ranked document set, fitting a maximum likelihood Dirichlet distriubtion to the sampled feedback models and selecting the mode of this Dirichlet

as the combined feedback model. This stabilizes the feedback models produced by our baseline feedback algorithm, choosing terms that are consistent across multiple training sets, and thereby increasing the precision of the feedback models. We used self-organizing maps to give a novel visualization of how expansion model instability occurs in real query situations.

7.2.2 Query variant framework

We showed that sampling query variants is a way to help address poor initial retrieval quality, by increasing the number of query hypotheses. Each query variant perturbs the relative weights on the original query terms. In this way, even if the original query returns few or no relevant documents in the top 10 (say), the use of query variants increases the likelihood of finding at least a few more relevant documents. We discussed model combination approaches that are effective in combining the results of these different hypotheses. We introduced *sigma-point sampling*: the novel application of the unscented transform theorem from particle filtering to finding a good set of perturbation weights. We connected the idea of query perturbation to work on *local influence and sensitivity* in general statistics. We also introduced new measures to evaluate the effectiveness of a query expansion algorithm including the use of risk-reward curves.

7.2.3 Data perturbation kernels

We show that the use of query variants produces a valuable side benefit: training data for learning similarity measures over terms. We introduce data dependent kernels called *data perturbation kernels* and show how they can be derived theoretically from importance sampling methods on an integral (based on the Canonical Distortion Measure).

We applied data perturbation kernels in practical applications to both individual terms, and language models. When applied to terms, in addition to giving valuable risk and similarity data for convex optimization, it also induces a precise query-specific similarity measure between documents. We evaluated the effectiveness of the perturbation kernel as the term similarity measure in the QMOD covariance matrix for query expansion and compared this to a term association (Jaccard) measure with small but consistent gains. We also showed how treating the query as a random variable can be used to improve query difficulty prediction, generalizing an existing query difficulty measure by adding an interaction term for similarity between documents in a collection in addition to the similarity of the query

to the collection.

7.2.4 Optimization methods for query model estimation

In the final part of the thesis, we developed a convex optimization framework for estimating query models. This takes the view that the selection and weighting of query terms should be done as a set: that is, taking account the properties of the entire set of terms, instead of a greedy strategy that considers only the properties (such as relevance scores) that can result in aspect imbalance and other problems for smaller sets of terms. Another advantage of this approach is its simplicity. By reducing query model estimation to an optimization problem, it allows efficient general-purpose convex optimization techniques to be applied.

Our evaluation shows that a convex optimization approach provides further gains in robustness and resistance to noise over the heuristic methods of model combination in Chapter 3. The model combination method in Chapter 3 required a separate step to estimate whether or not to expand, whereas the convex program integrates everything into a single set of easily understood criteria, making changes and improvements much more transparent. In general, the use of an optimization framework provides a natural way to perform selective expansion, by constraining the objective with a feasible set of models that satisfy the conditions of reasonable expansion models, such as (possibly competing) constraints including aspect balance, term confidence, and coverage of query aspects.

There are a number of reasons that current Web search engines still do not use automatic feedback methods to increase result quality. Part of this thesis work has sought to address those reasons. First, there is great pressure to keep query processing times very fast (i.e. less than 250ms), not only to provide satisfying response times, but also to maximize user throughput for a site, and thus increase the number of advertisement views. The extra computation cost of automatic feedback is thus seen as a negative. Although our algorithms make use of more CPU time, we have tried to keep factors such as the number and nature of subqueries efficient. Second, some query expansion methods, like Latent Semantic Analysis [Deerwester et al. 1990], may operate by providing dozens or hundreds of weighted expansion terms. The reason why certain pages become highly ranked becomes somewhat more confusing and less predictable for a user. Our methods, in contrast, focus on choosing a small number of high-quality terms (including possibly no terms at all). Finally, if the average gains of automatic query model estimation are ever to be widely used to improve search, this downside risk must be greatly reduced. Using a principled convex optimization

approach, we have taken a significant step forward in solving this problem.

7.3 Future directions

The sampling and optimization approaches we have presented for information retrieval open new directions for research far beyond their use for query term weighting. We now discuss some possible generalizations to other domains.

Federated search under constraints. Much of the risk/reward analysis that is appropriate for query models can be applied to other types of information resources. Finding optimal query models can be seen as an instance of a much more general problem, of finding the value of one or more *information sources*, given a set of constraints on our access to them, and a description of relationships between the information sources. In federated search, for example, the information sources are not terms, but multiple databases or indexes. Given initial source quality or reliability statistics for each database, pair-wise overlap or similarity measures, and resource costs for access, we can apply a very similar convex optimization approach to solve for the information value weights for each collection. With these weights, we can prioritize the order in which collections are searched or their results ranked. Note that *sparse* solutions are likely to be even more important for federated search than for query expansion: the cost of a non-zero weight for a collection may imply the large overhead of searching a remote collection, so making the decision *not* to access one or more particular collections could be critical. If all indexes have no access cost, then a lossy approach may not be needed and redundant document removal can be done at merge time [Shokouhi & Zobel 2007]. However, to our knowledge little work in federated search addresses the problem of how to select or weight resources given constraints such as a fixed computation or access budget: if our access to resources is constrained, a lossy approach is unavoidable and an optimization problem must be solved[2].

Machine translation and summarization. Matching a query with a document can be viewed as a form of statistical translation [Berger & Lafferty 1999]. Conversely, methods we have developed in this thesis for improved matching, such as computing feedback term weights under constraints, may be useful for related tasks such as computing *n*-best candidate lists for statistical translation or cross-lingual information retrieval.

[2]We also note that the problem of constrained resource selection is closely related to the dual problem of information flow and capacity, where nodes are identified with edges or routes instead of indexes. In this case, the optimization problem can be viewed as allocating traffic amounts in the network.

Furthermore, instead of using our method to produce query models for feedback, we could apply it to individual documents to produce *query-biased summaries* of those documents. Such summaries would select a subset of terms, phrases, or sentences having balanced aspect coverage for the query. When the summarization problem has feasible solutions given the aspect coverage, term weight variance, and other constraints, we can choose a particular summary by specifying another criterion to optimize. For example, we can define the *most efficient summary* as the smallest set of terms that have adequate aspect coverage, i.e. that best predict the terms in the example document.

Extensions to semi-supervised and active learning. In general, we have sought to minimize our reliance on supervised learning methods. Training data is often difficult and/or expensive to obtain, and ad-hoc retrieval itself is more like a meta-learning problem that changes from query to query. Instead, we often create pseudo-training data for each individual query problem through the use of methods such as query variants. However, if training data is available for a particular task or user such as query classification or term similarity, it would be interesting to extend our methods to supervised learning. A resource like Web query logs may improve effectiveness by allowing more realistic modeling of the true task (query) neighborhood, which heavily affects the selection of related words. User feedback may also provide training data that can be used to constrain refinements of expansion term candidates. Another research direction related to semi-supervised learning is finding improved resampling strategies that exploit the clustering behavior of documents.

There are also interesting possibilities for active learning. One advantage of a convex optimization approach is that we can easily perform a sensitivity analysis of the constraints. That is, we can see how much the optimal query model is affected by small changes in each constraint. Then, we can focus on the constraints that affect the optimal solution the most, and potentially gather more information on those to refine the solution. For example, if the constraints include conditions on which expansion terms were marked as relevant by a hypothetical user, the sensitivity analysis would identify the most influential unlabelled terms, and thus find the most useful terms for which to obtain actual feedback. In this way, instead of using terms directly in an expanded query, we could maximize the utility we get from allowing the user to select and give us feedback. This process could be repeated: the observed selections could in turn be used to refine the list of expansion candidates by further constraints, such as requiring that the previously selected terms continue to have

high weight.

Optimizing over the domain of document weights also leads to the idea of using regularization of retrieval scores to reflect content similarity. Diaz introduced this concept recently by another path in his thesis [Diaz 2007a] and discusses cluster-based retrieval and its relation to score regularization. Instead of treating documents as independent sources, we can calculate the similarity between them and take this into account when estimating document weights for feedback. If we reward clusters, we can modify the variance constraint to prefer close weights with a cluster, but large differences between clusters. Alternatively, we can specify that the query model *diversify* its reliance on resources by emphasizing a single representative element from a cluster of similar resources.

Improving search by working harder. As we have shown with our results on bagging relevance models and running query variants, practical improvements in precision and robustness are directly achievable by increasing the CPU time available to the search engine to process a given query. This suggests an exciting direction for future research, namely, exploring how to improve search results further by *working harder*, either automatically or under the control of the user. Given the tremendous increase in cluster-based computing resources available for search and text mining, the tradeoff between computational complexity and retrieval effectiveness appears to hold some promise. Powerful operations like bagging and subquery retrieval are inherently parallelizable and, with an adaptive approach based on server load, the response time need not greatly extend the actual response time for the user. When CPU load is high, effort may simply be dynamically scaled back to the default simple search model. Further operations during a more CPU-intensive search, such as lazy evaluation of deeper linguistic concepts in specific documents, may also be part of a portfolio of more intensive methods. In this way, we hypothesize that significant performance gains may be possible for the most difficult types of queries.

Chapter 8

Bibliography

Bibliography

Aitchison, J. & Shen, S. M. [1980]. Logistic normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272.

Akaike, H. [1974]. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Akaike, H. [1979]. A bayesian extension of the minimum aic procedure of autoregressive model fitting. *Biometrika*, 66(2):237–242.

Al Mamunur Rashid, G. K. & Riedl, J. [2005]. Influence in ratings-based recommender systems: An algorithm-independent approach. In *SIAM International Conference on Data Mining*.

Alizadeh, F. & Goldfarb, D. [2001]. Second-order cone programming. Tech. Rep. RRR-51-2001, Rutgers University.

Amati, G., Carpineto, C. & Romano, G. [2004]. Query difficulty, robustness, and selective application of query expansion. In *Proceedings of the 25th European Conference on Information Retrieval (ECIR 2004)*, pages 127–137.

Anagnostopoulos, A., Broder, A. Z. & Carmel, D. [2005]. Sampling search-engine results. In *WWW '05: Proceedings of the 14th International Conference on the World Wide Web*, pages 245–256. ACM Press, New York, NY, USA. ISBN 1-59593-046-9.

Anderson, E. [1999]. Mosek ver 1.0b user's manual. URL `http://www.mosek.com`.

Ando, R. K., Dredze, M. & Zhang, T. [2006]. TREC 2005 genomics track experiments at IBM Watson. In *Proceedings of TREC 2005*. NIST Special Publication.

Ando, R. K. & Zhang, T. [2005]. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 1–9.

Aslam, J. A. & Pavlu, V. [2007]. Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR 2007)*, pages 198–209.

Bar-Yossef, Z. & Gurevich, M. [2006]. Random sampling from a search engine's index. In *WWW '06: Proceedings of the 15th International Conference on the World Wide Web*, pages 367–376.

Baxter, J. [1997]. The canonical distortion measure for vector quantization and function approximation. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 39–47.

Bennett, P. N. [2007]. Neighborhood-based local sensitivity. In *Proceedings of the European Conference on Machine Learning (ECML 2007)*, pages 30–41.

Berger, A. & Lafferty, J. [1999]. Information retrieval as statistical translation. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229. ACM, New York, NY, USA. ISBN 1-58113-096-1.

Billerbeck, B. [2005]. *Efficient Query Expansion*. Ph.D. thesis, RMIT University, Melbourne, Australia.

Blanzieri, E. & Ricci, F. [1999]. Probability based metrics for nearest neighbor classification and case-based reasoning. In *ICCBR '99: Proceedings of the Third International Conference on Case-Based Reasoning and Development*, pages 14–28. Springer-Verlag, London, UK. ISBN 3-540-66237-5.

Boyd, S. & Vandenberghe, L. [2004]. *Convex Optimization*. Cambridge University Press, New York, NY, USA. ISBN 0521833787. URL `http://www.stanford.edu/~boyd/cvxbook/`.

Breiman, L. [1996]. Bagging predictors. *Machine Learning*, 24(2):123–140.

Buckley, C. [2004]. Why current IR engines fail. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 584–585.

Carpineto, C., de Mori, R., Romano, G. & Bigi, B. [2001a]. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27.

Carpineto, C., Romano, G. & Giannini, V. [2001b]. Improving retrieval feedback with multiple term-ranking function combination. *ACM Transactions on Information Systems*, 20(3):259 – 290.

Collins-Thompson, K. & Callan, J. [2005]. Query expansion using random walk models. In *Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM 2005)*, pages 704–711.

Collins-Thompson, K. & Callan, J. [2007]. Estimation and use of uncertainty in pseudo-relevance feedback. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 303–310.

Collins-Thompson, K., Ogilvie, P. & Callan, J. [2004]. Initial results with structured queries and language models on half a terabyte of text. In *Proc. of 2005 Text REtrieval Conference*. NIST Special Publication.

Cook, R. D. [1979]. Influential observations in linear regression. *Journal of the American Statistical Association*, 74(2):169–174.

Cook, R. D. [1986]. Assessment of local influence. *Journal of the Royal Statistical Society B*, 48(2):133–169.

Cook, R. D. & Weisberg, S. [1982]. *Residuals and Influence in Regression*. Chapman and Hall. ISBN 041224280X. URL `http://www.stat.umn.edu/rir/rir.pdf`.

Crabtree, D., Andreae, P. & Gao, X. [2007]. Exploiting underrepresented query aspects for automatic query expansion. In *Knowledge Discovery and Data Mining Conference (KDD 2007)*, pages 191–200.

Croft, W. B. & Thompson, R. H. [1987]. I3r: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6):389–404.

Cronen-Townsend, S. & Croft, W. [2002]. Quantifying query ambiguity. In *Proceedings of HCL 2002*, pages 94–98.

Crouch, C. J., Crouch, D. B., Chen, Q. & Holtz, S. J. [2002]. Improving the retrieval effectiveness of very short queries. *Information Processing and Management*, 38(1):1–36. ISSN 0306-4573.

Davis, J. & Goadrich, M. [2006]. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 233–240.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G. & Harshman, R. [1990]. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology (JASIST)*, 41(6):391–407.

Diaconis, P. & Holmes, S. [1994]. Gray codes for randomization procedures. *Statistics and Computing*, 4(4):287–302.

Diaz, F. [2007a]. *Autocorrelation and Regularization of Query-Based Information Retrieval SCores*. Ph.D. thesis, University of Massachusetts, Amherst.

Diaz, F. [2007b]. Performance prediction using spatial autocorrelation. In *Proceedings of the 30th Annual Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 583–590.

Dillon, J., Mao, Y., Lebanon, G. & Zhang, J. [2007]. Statistical translation, heat kernels, and expected distances. In *Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI 2007)*.

Doyle, L. B. [1961]. Semantic road maps for literature searchers. *Journal of the ACM*, 8(4):553–578. ISSN 0004-5411.

Duda, R. O., Hart, P. E. & Stork, D. G. [2001]. *Pattern Classification*. Wiley and Sons, 2nd ed.

Efron, B. [1979]. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7:1–26.

Fazel, M. [2002]. *Matrix Rank Minimization with Applications*. Ph.D. thesis, Stanford University.

Ghahramani, Z. & Kim, H.-C. [2003]. Bayesian classifier combination. Tech. rep., Gatsby Technical Report.

Gibas, M., Zheng, N. & Ferhatosmanoglu, H. [2007]. A general framework for modeling and processing optimization queries. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Databases*, pages 1069–1080. VLDB Endowment. ISBN 978-1-59593-649-3.

Greiff, W. R., Morgan, W. T. & Ponte, J. M. [2002]. The role of variance in term weighting for probabilistic information retrieval. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM 2002)*, pages 252–259. ISBN 1-58113-492-4.

Harman, D. & Buckley, C. [2004]. The NRRC Reliable Information Access (RIA) workshop. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 528–529. ACM, New York, NY, USA. ISBN 1-58113-881-4.

J. Peng, S. Q., B. Bhanu [1999]. Probabilistic feature relevance learning for content-based image retrieval. *Computer Vision and Image Understanding*, 75(12):150–164. Academic Press.

Jaakkola, T., Diekhans, M. & Haussler, D. [1999]. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158. AAAI Press. ISBN 1-57735-083-9.

Jebara, T., Kondor, R. & Howard, A. [2004]. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844. ISSN 1533-7928.

Jones, K. S. [1971]. *Automatic keyword classification for information retrieval*. Butterworths.

Julier, S. & Uhlmann, J. K. [2002]. The scaled unscented transformation. *Proceedings of the IEEE American Control Conference*, pages 4555–4559.

Kalos, M. H. & Whitlock, P. A. [1986]. *Monte Carlo Methods: Volume 1*. Wiley and Sons, New York, NY.

Karger, D. [1994]. *Random Sampling in Graph Optimization Problems*. Ph.D. thesis, Stanford University.

Kohonen, T., Hynninen, J., Kangas, J. & Laaksonen, J. [1996]. SOMPAK: The self-organizing map program package. Tech. Rep. A31, Helsinki University of Technology. URL `http://www.cis.hut.fi/research/papers/som_tr96.ps.Z`.

Kurland, O., Lee, L. & Domshlak, C. [2005]. Better than the real thing?: Iterative pseudo-query processing using cluster-based language models. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–26. ACM, New York, NY, USA. ISBN 1-59593-034-5.

Lafferty, J. D. & Lebanon, G. [2002]. Information diffusion kernels. In *Advances in Neural Information Processing Systems (NIPS) 15*, pages 375–382. MIT Press.

Lafferty, J. D. & Zhai, C. [2001]. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119.

Lanckriet, G., El Ghaoui, L., Bhattacharyya, C. & Jordan, M. [2002]. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582.

Lavrenko, V. [2004]. *A Generative Theory of Relevance*. Ph.D. thesis, University of Massachusetts, Amherst.

Lee, C.-H., Greiner, R. & Wang, S. [2006]. Using query-specific variance estimates to combine Bayesian classifiers. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 529–536.

Lee, K.-S., Croft, W. B. & Allan, J. [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*, pages 235–242.

Lee, L. [1999]. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32. Association for Computational Linguistics, Morristown, NJ, USA. ISBN 1-55860-609-3.

Lemur [2002]. The Lemur toolkit for language modeling and retrieval. URL `http://www.lemurproject.org`.

Lobo, M. S., Fazel, M. & Boyd, S. [2007]. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 152(1):376–394.

Manning, C. & Schütze, H. [2000]. *Foundations of Statistical Natural Language Processing*. MIT Press. ISBN 0-262-13360-1.

Markowitz, H. M. [1952]. Portfolio selection. *Journal of Finance*, 7(1):77–91.

McCulloch, R. E. [1989]. Local model influence. *Journal of the American Statistical Association*, 84(406):473–478.

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

Metzler, D., Dumais, S. T. & Meek, C. [2007]. Similarity measures for short segments of text. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*, pages 16–27.

Minka, T. [2000a]. Distance measures as prior probabilities. Tech. rep. URL `http://research.microsoft.com/~minka/papers/metric`.

Minka, T. [2000b]. Estimating a Dirichlet distribution. Tech. rep. URL `http://research.microsoft.com/~minka/papers/dirichlet`.

Mitra, M., Singhal, A. & Buckley, C. [1998]. Improving automatic query expansion. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214. ACM, New York, NY, USA. ISBN 1-58113-015-5.

Pittman, J. [1993]. *Probability*. Springer-Verlag, New York.

Ponte, J. [2000]. *Advances in Information Retrieval*, chap. Language models for relevance feedback, pages 73–96. W.B. Croft, ed.

Ponte, J. M. & Croft, W. B. [1998]. A language modeling approach to information retrieval. In *Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281.

Prabhu, G. M. & Deo, N. [1984]. On the power of perturbation for testing non-isomorphism of graphs. *BIT*, 24:302–307.

Preece, S. E. [1973]. Clustering as an output option. *Proceedings of the American Society for Information Science*, 10:189–190.

Qiu, Y. & Frei, H. [1993]. Concept-based query expansion. In *SIGIR 1993: Proceedings of the 16th annual international ACM SIGIR conference on information retrieval*, pages 160–170. ACM Press, New York, NY, USA.

Ravikumar, P. & Lafferty, J. [2006]. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 737–744.

Resnik, P. [1995]. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of the 14th international joint conference on artificial intelligence (IJCAI)*, pages 448–453.

Robertson, S. E. [1977]. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304.

Sahami, M. & Heilman, T. [2006]. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the Fifteenth International World Wide Web Conference (WWW 2006)*, pages 377–386.

Sakai, T., Manabe, T. & Koyama, M. [2005]. Flexible pseudo-relevance feedback via selective sampling. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(2):111–135. ISSN 1530-0226.

Salton, G., Fox, E. & Wu, H. [1983]. Extended boolean information retrieval. *Communications of the ACM*, 26(1):1022–1036.

Shokouhi, M. & Zobel, J. [2007]. Federated text retrieval from uncooperative overlapped collections. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 495–502. ACM, New York, NY, USA. ISBN 978-1-59593-597-7.

Short, R. & Fukanaga, K. [1980]. A new nearest neighbor distance measure. In *International Conference on Pattern Recognition 1980*, pages 81–86.

Strohman, T. [2007]. Private communication.

Strohman, T., Metzler, D., Turtle, H. & Croft, W. B. [2004]. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*.

Sturm, J. F. [2004]. Sedumi: optimization over symmetric cones. McMaster University, Hamilton, ON. Canada. URL `http://sedumi.mcmaster.ca/`.

Swets, J. A. [1963]. Information retrieval systems. *Science*, 141:245–250.

Tao, T. & Zhai, C. [2006]. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 2006 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 162–169.

Teevan, J., Dumais, S. T. & Horvitz, E. [2005]. Personalizing search via automated analysis of interests and activities. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 449–456. ACM, New York, NY, USA. ISBN 1-59593-034-5.

Thompson, R. & Croft, W. B. [1989]. Support for browsing in an intelligent text retrieval system. *International Journal of Man Machine Studies*, 30:639–668.

Tobin, J. [1956]. Liquidity preference as behavior towards risk. *Cowles Foundation Discussion Papers*, (14). URL `http://ideas.repec.org/p/cwl/cwldpp/14.html`.

Tombros, A. & van Rijsbergen, C. J. [2001]. Query-sensitive similarity measures for the calculation of interdocument relationships. In *Proceedings of the Tenth International Conference on Information and Knowledge management (CIKM 2001)*, pages 17–24. ACM, New York, NY, USA. ISBN 1-58113-436-3.

Tsuda, K. & Kawanabe, M. [2002]. The leave-one-out kernel. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 727–732. Springer-Verlag, London, UK. ISBN 3-540-44074-7.

Tsuda, K., Kin, T. & Asai, K. [2002]. Marginalized kernels for biological sequences. *Bioinformatics*, 18(1):268–275.

Tsuda, K. & Ratsch, G. [2005]. Image reconstruction by linear programming. *IEEE Transactions on Image Processing*, 14(6):737–744. ISSN 1057-7149.

van Rijsbergen, K. [1979]. *Information Retrieval*. Butterworths, London.

Vandenberghe, L. [2008]. Cvxopt: A python package for convex optimization. URL `http://abel.ee.ucla.edu/cvxopt`.

Veach, E. [1997]. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. thesis, Stanford University.

Vinay, V., Cox, I. J., Milic-Frayling, N. & Wood, K. [2005]. On ranking the effectiveness of searches. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 398–404.

Voorhees, E. & Harman, D. [2005]. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, Cambridge, MA. ISBN 0-262-22073-3.

Watkins, C. [2000]. *Advances in Large Margin Classifiers*, chap. Dynamic Alignment Kernels, pages 39–50. MIT Press.

Willett, P. [1985]. Query specific automatic document classification. *International Forum on Information and Documentation*, 10(2):28–32.

Winaver, M., Kurland, O. & Domshlak, C. [2007]. Towards robust query expansion: model selection in the language modeling framework. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 729–730. ACM, New York, NY, USA. ISBN 978-1-59593-597-7.

Xu, J. & Croft, W. B. [1996]. Query expansion using local and global document analysis. In *Proceedings of the 1996 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11.

Xu, J. & Croft, W. B. [2000]. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1):79–112. ISSN 1046-8188.

Yianilos, P. [1995]. Metric learning via normal mixtures. Tech. rep. URL `citeseer.ist.psu.edu/yianilos95metric.html`.

Yih, W. & Meek, C. [2007]. Improving similarity measures for short segments of text. In *The Twenty-First AAAI Conference on Artificial Intelligence*, pages 1489–1494.

YomTov, E., Fine, S., Carmel, D. & Darlow, A. [2005]. Learning to estimate query difficulty. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–519.

Zhai, C. & Lafferty, J. [2006]. A risk minimization framework for information retrieval. *Information Processing and Management*, 42(1):31–55. ISSN 0306-4573.

Zhai, C. X., Cohen, W. W. & Lafferty, J. [2003]. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR '03: Proceedings of the 26th An-*

nual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 10–17. ACM, New York, NY, USA. ISBN 1-58113-646-3.

Zhao, Q., Bhowmick, S., Hoi, S. C. H., Lyu, M. R., Liu, T.-Y. & Ma, W.-Y. [2006]. Time-dependent semantic similarity measure of queries using historical click-through data. In *WWW '06: Proceedings of the 15th International Conference on the World Wide Web*. ISBN 1-59593-323-9/06/0005.

Zhou, Y. & Croft, W. B. [2006]. Ranking robustness: a novel framework to predict query performance. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006)*, pages 567–574. ISBN 1-59593-433-2.

Appendix A

Background Material

This Appendix contains background definitions and explanation of important concepts used in the dissertation.

A.1 Kullback-Leibler divergence

Given probability mass functions $p(x)$ and $q(x)$, the Kullback-Leibler divergence between p and q is defined as

$$KL(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{A.1}$$

The Kullback-Leibler divergence is typically referred to as *KL-divergence* or sometimes as *relative entropy*. Note that $KL(p||q)$ is not symmetric: choosing $KL(q||p)$ instead of $KL(p||q)$ can lead to very different results in some situations. Thus, $KL(p||q)$ computes the similarity of distributions, but is not a true 'distance' metric between them (and the triangle equality also does not hold). $KL(p||q)$ is zero if and only if p and q are identical distributions, and is always non-negative and is undefined if $q(x) = 0$ for any x.

Jensen-Shannon divergence is a symmetrized and smoothed version of KL-divergence, defined as

$$JS(p||q) = \frac{1}{2}KL(p||m) + \frac{1}{2}KL(q||m) \tag{A.2}$$

where

$$m(x) = \frac{1}{2}(p(x) + q(x)). \tag{A.3}$$

The JS-divergence is the average of KL-divergences to the average distribution. While $JS(p||q)$ does not define a metric, $\sqrt{JS(p||q)}$ does give a metric called the *Hellinger metric*.

A.2 Association measures

If we have two binary attributes A and B, we can define various *association measures* in terms of the four cells M_{ab} in a 2×2 contingency table where M_{11} is the number of samples with both $A = 1$ and $B = 1$, M_{10} is the number of samples with $A = 1$ and $B = 0$, and so on. For query expansion, the binary event we use for word w_A corresponds to a document containing w_A, resulting in *term association measures*.

One association measure, the *Jaccard coefficient* $\Delta_{JACCARD}$ is given by

$$\Delta_{JACCARD} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}. \tag{A.4}$$

This gives a similarity measure between 0 and 1. We convert this into a distance measure $\delta_{JACCARD}$ compatible with the perturbation kernel by rescaling using the formula

$$\delta_{JACCARD}(w_a, w_b) = \beta_1 \exp -\beta_2 \cdot \Delta_{JACCARD}(w_a, w_b) \tag{A.5}$$

where $\beta_1 = 15.0$ and $\beta_2 = 2.0$, which were obtained using empirical tuning.

Another association measure used in our experiments is *Yule's Q Coefficient*, which is defined as

$$Q = \frac{OR - 1}{OR + 1} \tag{A.6}$$

where OR is the odds-ratio

$$OR = \frac{M_{00}M_{11}}{M_{01}M_{10}}. \tag{A.7}$$

In practice we use a rescaled variant $\hat{Q}$ defined as

$$\hat{Q} = \frac{1}{2}(1 + Q). \tag{A.8}$$

Appendix B

Statistical measures of influence

The idea of performing tests to measure the sensitivity of a model to perturbations in the training data or model parameters has been of interest to statisticians for some time. Initial attempts at sensitivity analysis in the 1970s proposed diagnostics that focused on perturbing the case-weights for the simple scenario of linear regression [1]. The seminal work of Cook resulting in Cook's D-statistic [Cook 1979] is an example of one such measure. Cook's D-statistic estimates the influence of a single case (the *i*-th data element) on the model when the case is left out of the training set. The influence D_i of case *i* is given as

$$D_i = \frac{||\hat{Y} - \hat{Y}_{(i)}||^2}{p\dot{\sigma}^2} \tag{B.1}$$

where $\hat{Y}$ and $\hat{Y}_{(i)}$ are the $nx1$ vectors of fitted values on the full training set, and without case *i*, respectively. Also, *p* is the dimension of the parameter space β (where $Y = X \cdot \beta + \epsilon$, where ϵ is Gaussian noise.) Cook and Weisberg [Cook & Weisberg 1982] later extended this to leaving out a subset of the data.

In a later paper, Cook [Cook 1986] further generalized this idea by introducing *local influence* methods that are closely connected with the query perturbation we describe later. Given observed data *y* and a *q*-dimensional perturbation vector ω he defined a likelihood

[1] In the statistics literature, what we might call a training point, namely, an observation on a response variable in combination with values for the explanatory variables, is termed a *case*. The *case-weights* are what we have termed training set weights. Estimators such as leave-one-out estimators that set case-weights to zero are termed *case deletion* schemes [Cook 1986].

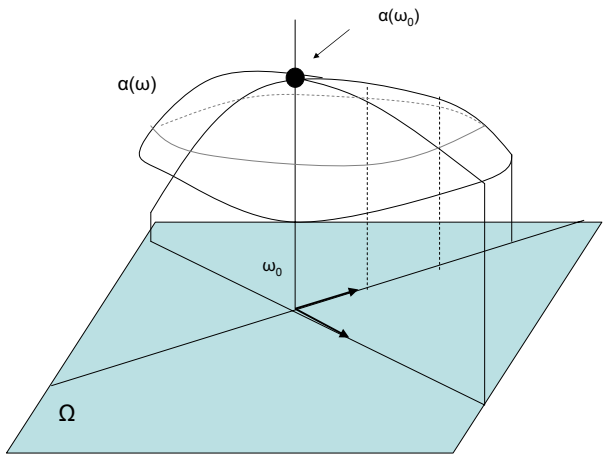


Figure B.1: By varying ω in the space Ω, a surface $\alpha(\omega)$ is generated. Local influence measures the curvature of $\alpha(\omega)$ at the point ω_0.

displacement (LD)

$$LD(\omega) = 2 \cdot (L(\hat{\theta}, y) - L(\hat{\theta_\omega})) \tag{B.2}$$

as a way to measure the distance of the parameter estimates between perturbed and unperturbed responses. Here, $\hat{\theta}$ and $\hat{\theta_\omega}$ are the maximum-likelihood estimates for θ in the unperturbed and perturbed models respectively.

Using these *LD* values, Cook then defined *influence graphs*, which summarize the influence of a perturbation scheme ω on a model with parameters θ and log-likelihood function $L(\theta)$. The influence graph is the geometric surface formed as ω varies in the space Ω, giving the $q + 1$ vector.

$$\alpha(\omega) = \begin{pmatrix} \omega \\ LD(\omega) \end{pmatrix}. \tag{B.3}$$

The vector $\omega_0 = \mathbf{1}$ (all ones) represents no change to the data, with all points getting a weight of 1.0. Applying concepts from differential geometry, Cook defined the *local influence diagnostic* as the direction of maximum curvature on the influence graph around the point ω_0. The goal is to summarize how the surface $\alpha(\omega)$ deviates from its tangent plane at ω_0. This can be done by examining the curvature of specifically-chosen curves called *normal sections* on $\alpha(\omega)$ that pass through $\alpha\omega_0$. This is illustrated in Figure B.1.

McCulloch [McCulloch 1989] develops a Bayesian version of these ideas: a sensitivity analysis that measures the change in the posterior distribution given changes in either the sampling distribution or the prior distribution. The *Fisher information* matrix

$$G(\omega) = \frac{\partial^2 k(\omega)}{\partial \omega_i \ldots \partial \omega_j} \tag{B.4}$$

is used to form the statistic

$$\lambda^\star = \max_{\|\delta\|=1} \frac{\delta^\mathsf{T} G_{POST}(\omega_0)\delta}{\delta^\mathsf{T} G_{PRIOR}(\omega_0)\delta} \tag{B.5}$$

The eigenvector $\delta^\star$ corresponding to the largest eigenvalue $\lambda^\star$ gives valuable information about the perturbations that achieve the largest local change in $LD(\omega)$, enabling us to obtain the relative importance of the elements of ω. This is useful diagnostic information on the sensitivity of the model. The largest absolute elements of L_{max} correspond to the cases (training points) in data having the largest influence on the posterior distribution.

Appendix C

TREC Evaluation

This section summarizes the TREC corpora and topics used in our experiments. These datasets provide a standardized methodology for comparing the performance of different retrieval algorithms. A TREC evaluation set consists of three parts: topics, collections, and human relevance judgments. These are supplied by the Information Retrieval Laboratory at the U.S. National Institute of Standards (NIST). This data may be downloaded from the TREC site at NIST: `http://trec.nist.gov`. Further information on the TREC assessment and evaluation methodology may be found in [Voorhees & Harman 2005].

Information needs in TREC are expressed as *topics*. Typically, a topic comprises a short, medium, and long description need. These are called, respectively, the title, description, and narrative fields. An example of a TREC topic is given in Figure C.1. An information retrieval system takes a set of topics as input, converting each topic to the appropriate query form for that system. The system runs the query on the test collection, returning a ranked list of documents. Each document has a document ID that was provided as part of its entry in the collection. Because the human relevance judgments use the same document IDs, the system's results may be scored for relevance against the human judgments, thus enabling us to calculate standard IR measures for the system, such as precision, recall, etc.

Table C.1 gives summary statistics for the four TREC collections used in this thesis. Some topic sets use the same underlying collection: for example, the topic sets for TREC 7, TREC 8, and Robust 2004 all use the collection built from the content on TREC Disks 4&5 (minus the Congressional Record documents). In such cases, the different topic set names and numeric ranges are given on the same row, separated by semi-colons. The relevance judgements provided by NIST are compiled by human assessors. These are stored in a

Collection	Description	Docs	Size	Avg Doc Len	Topics
TREC 1&2	Newswire articles (TREC Disks 1&2)	741,856	2.099Gb	2.967Kb	51–150, 151–200
TREC 7; TREC 8; Robust 2004	Newswire articles (TREC Disks 4&5 minus CR)	527,094	1.36Gb	474 bytes	351–400; 401–450; 301–450, 601–700
WT10g	Small web crawl	1,692,096	10Gb	6.2Kb	451–550
GOV2 (2004-2006)	Crawl of .gov domain	25,205,179	400Gb	15.0Kb	701–850

Table C.1: Summary statistics for TREC collections used in this thesis.

```
<top>
<num> Number:  701

<title> U.S. oil industry history

<desc> Description:
Describe the history of the U.S. oil industry

<narr> Narrative:
Relevant documents will include those on historical exploration and
drilling as well as history of regulatory bodies.  Relevant are history
of the oil industry in various states, even if drilling began in 1950
or later.

</top>
```

Figure C.1: Example of a TREC topic (topic 701), showing the short, medium, and long descriptions of an information need.

```
701 0 GX000-00-13923627 0
701 0 GX000-13-3889188 0
701 0 GX000-15-11323601 0
701 0 GX000-21-7072501 0
701 0 GX000-22-11749547 0
701 0 GX000-25-2008761 1
701 0 GX000-27-14827260 0
701 0 GX000-27-4783281 0
701 0 GX000-41-2972136 0
701 0 GX000-43-8149041 2
701 0 GX000-45-2286833 0
701 0 GX000-46-2808962 0
701 0 GX000-48-10208090 0
701 0 GX000-55-12164304 0
701 0 GX000-55-3407826 2
701 0 GX000-67-12045787 2
```

Figure C.2: Sample TREC relevance assessment format, showing the first few assessments for Topic 701

qrels file. An example of the format for a TREC qrels file is shown in Figure C.2. The fields are, in order: the topic number, an unused field, document ID, and relevance score. The relevance score takes values of 0 (not relevant), 1 (relevant), or 2 (highly relevant).

Index