A LANGUAGE MODELING APPROACH TO INFORMATION RETRIEVAL

A Dissertation Presented

by

JAY MICHAEL PONTE

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

DOCTOR OF PHILOSOPHY

September 1998

Department of Computer Science

A LANGUAGE MODELING APPROACH TO INFORMATION RETRIEVAL

A Dissertation Presented

by

JAY MICHAEL PONTE

Approved as to style and content by:

W. Bruce Croft, Chair

Andrew G. Barto, Member

James Allan, Member

Ramesh Korwar, Member

James Kurose, Department Chair
Department of Computer Science

ABSTRACT

A LANGUAGE MODELING APPROACH TO INFORMATION RETRIEVAL

SEPTEMBER 1998

JAY MICHAEL PONTE

B.S., NORTHEASTERN UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

In today's world, there is no shortage of information. However, for a specific information need, only a small subset of all of the available information will be useful. The field of information retrieval (IR) is the study of methods to provide users with that small subset of information relevant to their needs and to do so in a timely fashion. Information sources can take many forms, but this thesis will focus on text based information systems and investigate problems germane to the retrieval of written natural language documents.

Central to these problems is the notion of "topic." In other words, what are documents about? However, topics depend on the semantics of documents and retrieval systems are not endowed with knowledge of the semantics of natural language. The

approach taken in this thesis will be to make use of probabilistic language models to investigate text based information retrieval and related problems.

One such problem is the prediction of topic shifts in text, the topic segmentation problem. It will be shown that probabilistic methods can be used to predict topic changes in the context of the task of new event detection. Two complementary sets of features are studied individually and then combined into a single language model. The language modeling approach allows this problem to be approached in a principled way without complex semantic modeling.

Next, the problem of document retrieval in response to a user query will be investigated. Models of document indexing and document retrieval have been extensively studied over the past three decades. The integration of these two classes of models has been the goal of several researchers but it is a very difficult problem. Much of the reason for this is that the indexing component requires inferences as to the semantics of documents. Instead, an approach to retrieval based on probabilistic language modeling will be presented. Models are estimated for each document individually. The approach to modeling is non-parametric and integrates the entire retrieval process into a single model. One advantage of this approach is that collection statistics, which are used heuristically for the assignment of concept probabilities in other probabilistic models, are used directly in the estimation of language model probabilities in this approach. The language modeling approach has been implemented and tested empirically and performs very well on standard test collections and query sets.

In order to improve retrieval effectiveness, IR systems use additional techniques such as relevance feedback, unsupervised query expansion and structured queries. These and other techniques are discussed in terms of the language modeling approach and empirical results are given for several of the techniques developed. These results provide further proof of concept for the use of language models for retrieval tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

This thesis explores the use of probabilistic language models for information retrieval (IR). Many of the success stories in natural language processing including speech recognition, text extraction, part-of-speech tagging, and parsing have relied heavily on probabilistic methods in general and language models in particular. The work presented here is the development of an approach to information retrieval and related problems based on probabilistic language models.

The use of language models allows retrieval problems to be phrased in a manner that is conceptually simpler than many other approaches. Estimation of probabilities will play the key role in many of the techniques developed. This means that existing techniques for improving estimation can be brought to bear on the retrieval task.

This chapter begins with some basic concepts and preliminary definitions related to information retrieval in section 1.1. These definitions provide context for the discussion of retrieval models. After that, the concept of "topic," a central idea in information retrieval, will be discussed in section 1.2. The difficulties associated with that concept will also be discussed.

Next, the concept of language modeling in the context of information retrieval will be introduced, in section 1.3 and two applications will be introduced, interactive retrieval, discussed in section 1.3 and prediction of topic shifts, discussed in 1.4. Language modeling is a more tractable problem than topic identification and these applications serve to underscore that fact. The chapter concludes with the contribu-

tions that this thesis makes to the field of information retrieval, as well as an outline of the remainder of this document in Section 1.5.

1.1 Background Information and Preliminary Definitions

Some general background information and preliminary definitions about the study of information retrieval and related tasks will now be presented. The IR task is the retrieval of unstructured information. Such information might include text, images, audio etc., For the purposes of this thesis the discussion will be restricted to text based information systems. In the standard information retrieval task, documents are pre-defined and the retrieval system will retrieve documents in order to satisfy information needs of users. A related task is passage retrieval where the system retrieves only the most relevant portions of documents.

In a typical application, a collection of documents may be several gigabytes in size and will contain on the order of millions of documents. In a collection of this size, often only a few hundred documents, or fewer, will be relevant to a specific query. This large disparity in the cardinality of the relevant set of documents vs. the non-relevant set makes the problems quite different from many classification tasks and affects how retrieval systems are designed as well as how they are evaluated.

1.1.1 Information Retrieval

A collection of documents will be indexed by a set of features. In a text based retrieval system, features can include words, phrases etc. or manually assigned controlled vocabulary items. The focus of this dissertation will be automatic methods and so manually assigned features will not be considered.

When one has chosen a feature set, two approaches to retrieval are exact match methods and ranked methods (also know as partial match methods). For the sake of discussion, assume that the feature set consists of individual words. An exact

match method would return a set of documents corresponding to the strict Boolean combination of indexing features. For a collection of one million documents, it is highly unlikely that the relevant set can be retrieved correctly since the event space is the power set of the set of all documents in the collection. That is, one set of documents must be chosen out of $2^{1,000,000}$ possible sets. Moreover, this choice must be made on the basis of a single, relatively short, user query. On the other hand, for ranked retrieval, any ranking that tends to place relevant documents near the top of the list is useful.

Ranked retrieval methods retrieve a ranked list of documents rather than an unordered set. This allows these methods to take advantage of the fact that some features are better discriminators than others due to their occurrence statistics. Two useful measures for determining the importance of a feature are term frequency, *tf*, and inverse document frequency *idf*. The term frequency of a word is a function of the number of occurrences of that word in a given document – *tf* is a document level statistic. The inverse document frequency of a word is a function of the proportion of documents that a word occurs in – *idf* is a collection level statistic.

This information, combined in an appropriate fashion, can provide a very useful ranking of documents, typically ranking several relevant documents near the top of the list. With exact match systems, many documents, both relevant and non-relevant will typically be retrieved due to the number of possible sets of documents. Since these documents are presented to the user in no particular order, the user will be left to sift through a large quantity of documents to find the relatively few interesting ones.

For this reason, partial match retrieval is a better solution to the problem of document retrieval and will be the focus of this thesis. It should be pointed out that some users prefer Boolean systems because they feel as though they have more control of the search [11]. This issue will be addressed further in section 7.2.

There are two widely used classes of ranked retrieval models in IR, probabilistic models [46] and the vector space model [51] and the meaning of the ranking is different in each. Most probabilistic models follow the probability ranking principle [46]. This means that documents are ranked according to the probability of being relevant to the information need of the user. The retrieval model developed in Chapter 4 will be a departure from this. An example of a probabilistic retrieval model that follows the probability ranking principle is the INQUERY inference network model [61] based on Bayesian networks. This model as well as other probabilistic models are discussed in Chapter 2.

1.1.2 The Vector Space Model

The vector space model [51] treats documents and queries as vectors in an N-dimensional space, where N is the number of indexing features. The document vectors are ranked according to their cosine similarity (or, in general, according to some distance function) to the query vectors in that space. The major difference between probabilistic models and the vector space model is that while the probabilistic approaches attempt to explain the data by means of probabilistic modeling, the vector space model is more of an abstraction of the retrieval task itself. This means that in the vector space model the weighting and combination functions are determined empirically, rather than being prescribed or explained by the model. The semantics of the space are not defined, and the researcher is free to choose among distance functions. The tradeoff between the two approaches is that the vector space model is not burdensome to maintain, since it does not prescribe how retrieval should be done. On the other hand, the vector space model does not provide guidance to the researcher as to how retrieval performance might be improved as probabilistic models do.

1.1.3 Retrieval Systems

Information retrieval systems prepare a collection of documents for retrieval by the process of indexing. At indexing time, documents are tokenized into words and an inverted index file is built. The inverted file has a list for each indexing feature showing the documents that contain it. Phrases and proximity features can be calculated at retrieval time if the inverted file contains within-document positions of words. This approach allows more complex features, such as several words in proximity, without an explosion in index size, but incurs the cost of merging the ranked lists during query processing in order to calculate these features.

There are additional steps that may take place either at indexing time or at retrieval time. Each word will usually be compared against a stopword list. Stopwords are words with no discriminatory value such as "a", "an" and "the." These words are often discarded at indexing time in order to save space, although it is possible to perform stopping at retrieval time. An additional step is a form of morphological analysis, called stemming, that reduces variant forms of the same word to a common root form. For example "car" and "cars" would both be conflated to the same root. Stemming can also be performed at retrieval time by adding the variant word forms to the query [34].

1.1.4 Relevance Feedback

Relevance feedback is a technique for improving retrieval effectiveness based on relevance judgments provided by the user. At retrieval time, the user of the retrieval system poses a query, and the system retrieves a ranked list of documents. At this point, the user can provide relevance judgments about one or more documents in the ranked list. The retrieval system can modify the original query based on these relevance judgments. This is typically done by adding new terms from relevant documents and/or assigning weights to query terms based on occurrence statistics in the

judged documents. This modified query will then be run and a new ranked list will be returned. Relevance feedback typically provides a significantly improved ranking as compared to retrieval based on the initial query. Existing methods of relevance feedback will be covered in Chapter 2.

1.1.5 Information Routing/Filtering

The information routing or filtering task is similar to the retrieval task with relevance feedback. However, instead of a static collection of documents, the data is a dynamic stream of documents such as an online newswire. Each document is compared to a set of long-term user profiles and will be routed accordingly.

Since routing profiles represent long-term information needs, it is possible to collect a large number of relevance judgments over time, more judgments than one would expect for relevance feedback in the retrieval task. Much of the research in routing centers on how best to use these relevance judgments. This problem will be discussed further in Chapter 2 and again in Chapter 5.

1.2 Topics and Language Models

The techniques developed in this thesis, as well as problems in information retrieval in general, are intimately related to the concept of "topic." Yet, this concept is difficult to define. Retrieving documents relevant to the information need of the user can be viewed as a problem of inferring the topic that the user had in mind from the query, and making similar inferences about the documents in the collection. The documents could then be ranked according to the degree of belief that the document describes a similar topic to that which the user had in mind. This is essentially what existing probabilistic retrieval models do. An indexing model, one component of a probabilistic retrieval model, infers which concepts are likely to be present in the document, as will be described in Section 1.3. Additional inferences are made as to

the content of the query. In Section 1.1, interactive ranked retrieval was introduced. Probabilistic retrieval models attempt to formalize the process of ranked retrieval by interpreting the ranking scores as the probability that a document is relevant to the query. Relevance depends on user judgments, but one would expect users to judge as relevant those documents that described the topic expressed by the query.

The topic of the query is, however, a semantic notion, and retrieval systems work without semantic information. The important distinction to be made is the *de dicto* vs. the *de re* view of documents and queries, i.e., regarding the expression of documents and queries vs. regarding the physical objects expressed by documents and queries. To put this another way, should a retrieval model describe documents and queries or, should it describe what documents and queries describe? Existing approaches to retrieval make complex inferences as to the semantic content of documents and queries, i.e., the models make inferences as to what documents and queries describe – their subject matter or topic.

On what basis are these inferences made? A fortunate fact about language is that words are useful features for retrieval and, at the same time, carry considerable semantic content. (This is not true, for example, of image retrieval where distinguishing features, such as color histograms, do not correspond to what typical users would think of as the semantics of the image.) That words are useful features that carry semantic content helps make text retrieval an especially useful technology. Users can choose features that are meaningful to them and that will be useful to the system to find documents. Many existing models of retrieval explicitly make this connection. The task of retrieval is viewed as one of inference about semantic content based on word occurrences.

However, without deep understanding of natural language on the part of the retrieval system, the semantics are inferred from the words (or more complex features) that the query and the document have in common and, possibly, by addi-

tional collection-wide co-occurrence statistics. Subtleties of language such as figures of speech, ambiguities etc., are not accounted for by such a model, except to the extent that occurrence statistics preserve this information. The question becomes whether it is desirable to maintain this extra complexity as part of the model. If semantics are not being modeled explicitly, is it desirable to state that documents containing similar words are likely to be about similar topics, or that documents containing the same words as a query are likely to be relevant to the underlying topic expressed by that query? The view taken here is that this complexity is not necessary. Moreover, attempting to make inferences about the topic of documents has required researchers to estimate probabilities by heuristic methods.

The modeling of semantics requires the introduction of hidden variables. For example, in the 2-Poisson indexing model (described in Section 1.3) words are assumed to follow Poisson distributions with two different rates: one rate for documents that should have the word assigned as an indexing feature, and another rate for those documents for which the word should not be assigned. Whether a word should or should not be assigned to a document depends on whether a user posing that word as a query would be satisfied with the document. The mixture needs to be inferred since one cannot directly observe the phenomenon of interest.

Hidden variables are not necessarily a problem in and of themselves, and in some cases can be very appropriate. For example, in the speech recognition community, Hidden Markov Models (HMMs) are used to estimate the probability of words from acoustic signals. It is known that in human speech, sounds form equivalence classes which can be thought of as abstract sounds or "phonemes." In speech, an underlying phoneme will be expressed as its surface form (sometimes referred to as a "phone"). This process is governed by simple, well understood rules. Moreover, a phone is only one step removed from its underlying phoneme. For this reason, modeling a sequence of phonemes is a reasonable idea and the estimation of the probabilities is relatively

simple. On the other hand, the meaning of a document is several levels removed from the surface form and the rules of producing the surface form according to the underlying form are not entirely known. That being the case, should we believe that the probabilities of a hidden variable model can be estimated with any degree of accuracy?

1.2.1 Language Models

If one does not wish to model semantics what can be done instead? In this thesis, the concept of "topic" will be replaced by the concept of language model. In general, a language model is a probability distribution over strings in a finite alphabet. These probabilities can be estimated by a variety of techniques using as much (or as little) knowledge as one has about the language generation process. For example, suppose the alphabet consists of the two symbols '0' and '1.' In addition, suppose that the generation mechanism is known to be the following:

$$\begin{aligned} P(x_i = 0|x_{i-1} = 0) &= 0.9 \\ P(x_i = 0|x_{i-1} = 1) &= 0.1 \\ P(x_i = 1|x_{i-1} = 1) &= 0.9 \\ P(x_i = 1|x_{i-1} = 0) &= 0.1 \end{aligned}$$

These four probabilities can be viewed as a language model for the language in question, since they characterize the distribution of strings in this language. Another, simpler model for the above language is $P(x_i = 0) = P(x_i = 1) = 0.5$. This simpler model captures the distribution of symbols over the long term but does not account for the local sequential effects, i.e., it is a memoryless model. Also notice that the simpler model is vague in the sense that it describes any language consisting of an equal number of these two symbols regardless of the local sequential effects. This simpler model is analogous to the models that will be used for text retrieval.

The advantage of using language models is that the observable information, i.e., the collection statistics, can be used in a principled way to estimate these models and do not have to be used in a heuristic fashion to estimate the probability of processes that nobody fully understands. The view taken here then, is that the model should describe documents and queries themselves rather than make inferences as to their semantic content.

1.3 A Language Modeling Approach to Information Retrieval

The use of language models in the context of text retrieval allows for a simple formulation of the problem that is amenable to the application of many well studied techniques of probability estimation for the purposes of improving retrieval effectiveness. In addition, it will be shown that commonly used retrieval techniques, such as relevance feedback, have a natural interpretation using this approach.

The semantic inferences referred to in Section 1.2 are carried out by the component of a probabilistic retrieval model known as the indexing model, i.e., a model of the assignment of indexing terms to documents based on the concepts those terms represent. The discussion of indexing models will begin with a discussion of manual indexing, followed by the discussion of automatic indexing models in general. Finally, a specific example of an indexing model, the 2-Poisson model will be presented.

The notion of the indexing model can be thought of by analogy to manual indexing. In order to manually index a document, an expert indexer will read it, and assign words and phrases, often from a controlled indexing vocabulary. Manual indexers are highly trained and generally will have significant knowledge of the subjects of the documents they index. The assigned indexing terms in some way capture the content of the document in an abbreviated form and can then be used by searchers to find documents of interest.

The idea of manual indexing depends on two tacit assumptions:

- A small set of concepts can characterize the semantic content of a document.
- The indexing terms can be used to characterize these concepts.

However, the same document may be important to different people for different reasons. It is not likely that an indexer will be able to anticipate the information need of every user in advance. However, in a library environment, the concept of manual indexing often makes sense. The success of manual indexing depends on how well the indexers and the searchers agree on the assignment of the terms and on how well each is acquainted with the vocabulary of the subject (controlled or otherwise). In a library environment, where the indexers and the searchers have backgrounds in library science, or where the searchers at least have the assistance of librarians, one would hope that the agreement between searchers and indexers would be reasonably strong, and therefore, good retrieval performance would result.

Indexing models are an attempt to automate the process of index term assignment. Generally speaking, indexing models will weight terms that occur in the documents according to their suitability as indexing terms. Note that this is already a far cry from the manual indexing case where indexer and searcher are assumed to have considerable expertise and, in the best case, considerable agreement as to indexing terms. Moreover, information retrieval systems must retrieve documents in heterogeneous collections. These collections are considerably less "well behaved" than the text in a library environment and are likely to be less amenable to manual indexing in the first place.

In addition, the users of retrieval systems are typically not trained in library science and may not have a high degree of familiarity with the subject of documents for which they are searching. One might question whether the analogy to manual indexing is a reasonable basis at all for the model of a modern IR system.

This is especially the case when one considers that indexing models introduce additional complexity into retrieval models. More importantly, estimating the prob-

ability that a document is about a particular concept is very difficult. In practice, retrieval systems tend to use heuristic techniques to estimate these probabilities. The notion that a subset of important terms should be assigned to capture the meaning of a document is not warranted in the context of modern IR and, in practice, all of the words are used with the exception of stopwords. This being the case, modern indexing models have shifted emphasis away from the assignment of a select set of terms and, instead, try to weight every term. That means that every word is assumed to have an associated concept and the indexing model is meant to quantify to what extent a document is about a concept associated with each word in a document.

An early probabilistic indexing model is the 2-Poisson model, due to Bookstein and Swanson [9] and independently to Harter [28]. By analogy to manual indexing, the task was to assign a subset of words contained in a document (the "specialty words") as indexing terms. The probability model was intended to indicate the useful indexing terms by means of the differences in their rate of occurrence in documents "elite" for a given term, i.e., a document that would satisfy a user posing that single term as a query, vs. those without the property of eliteness.

The success of the 2-Poisson model has been somewhat limited because when one considers the full text in heterogeneous collections, the distribution of terms does not fit a mixture of two Poisson distributions very well [38]. In addition, documents were assumed to be of approximately equal length, a reasonable assumption for the data used in the initial studies [28] but not a reasonable assumption in general. To sum up, the 2-Poisson model makes several problematic assumptions that limit its usefulness, including:

- Inappropriate distributional assumptions.
- A subset of specialty words.
- A classification of documents into elite and non-elite sets.

- Uniform document length.

It should be noted that Robertson's *tf*, which has been shown to work very well empirically, was intended to behave similarly to the 2-Poisson model though no attempt was made by Robertson to fit a mixture of two Poisson distributions explicitly [48]. It appears that while Robertson's *tf* may have been inspired by the 2-Poisson model, it does not actually implement it.

Other researchers have proposed a mixture model of more than two Poisson distributions in order to better fit the observed data. Margulis proposed the n-Poisson model and tested the idea empirically [38]. The conclusion of this study was that a mixture of n-Poisson distributions provides a very close fit to the data. In a certain sense, this is not surprising. For large enough values of n, one can fit a very complex distribution arbitrarily closely by a mixture of n parametric models if one has enough data to estimate the parameters [59].

In any event, when one makes the leap from the 2-Poisson model to the n-Poisson model, the semantics of the mixture model are no longer clear, and so it is is not obvious how one would exploit the good fit even if it is meaningful.

Apart from the adequacy of the available indexing models, estimating the parameters of these models is a difficult problem. Researchers have looked at this problem from a variety of perspectives and several of these approaches will be discussed in Chapter 2. In addition, as previously mentioned, the current indexing models make two assumptions about the data that may be unwarranted.

- The parametric assumption.

- The assumption that semantics can be modeled by a latent variable.

In the approach developed in this thesis, these two assumptions will be relaxed. Rather than making parametric assumptions, as Silverman said, "the data will be

allowed to speak for themselves." [57] Unlike the current indexing models, the language modeling approach will not require the use of a hidden variable representing semantic content. Instead, rather than constructing a parametric model of the data, the actual data can be used by means of non-parametric methods.

Regarding the second assumption, the 2-Poisson model was originally based on the idea of "eliteness" [28]. It was assumed that a document elite for a given term would satisfy a user if the user posed that single term as a query. Since that time, the prevailing view has come to be that multiple term queries are more realistic. In general, this requires a combinatorial explosion of elite sets for all possible subsets of terms in the collection. The point of view taken here is that each query needs to be looked at individually and that documents will not necessarily fall cleanly into elite and non-elite sets. To sum up, in the manual indexing case, indexer and searcher are both experts with similar training. In the case of modern IR systems, the indexer is a machine and the user may have no training at all. The assumption that the semantics of documents can be captured by a simple latent variable model has not been shown to be reasonable in practice.

The approach that will be taken instead is based on probabilistic language modeling and integrates the entire retrieval process into a single model. The problematic notion of topic has been replaced with the notion of language model which can be used to capture the important features of documents in a useful way without making inferences about semantic content.

The approach to retrieval taken here is to infer a language model for each document and to estimate the probability of generating the query according to each model. The documents are then ranked according to these probabilities. This means that a single model describes the data, without the use of hidden concept variables, and the same model is used directly for retrieval.

The intuition behind this approach is that users have a reasonable idea of terms that are likely to occur in documents of interest and will choose query terms that distinguish these documents from others in the collection, an intuition discussed in more detail in section 7.2. When the task is stated this way, the view of the retrieval is that a model can capture the statistical regularities of text without inferring anything about the semantic content. This is very much in contrast to the indexing model approach, where inferences must be made about concepts that can be assigned to documents and that can be characterized by words.

Indexing models are based on the notion that there is a correct set of indexing terms that will characterize documents and allow users to find them. The view taken here is that there is no correct set of terms and that choosing indexing terms is successful only to the degree to which the indexer and user are in agreement, not to the degree to which the terms are correct. The task is no longer to estimate the probability that a given term is a correct indexing term and that will allow the model to be considerably simpler. This is very much in accord with the modern practice of indexing documents by all of the terms that occur in them (often with the exception of stopwords such as "the" which are usually not indexed).

By focusing on the query generation probability, as opposed to the probability of relevance, as in other models, the language modeling approach does not require a set of inferences for indexing and a separate set of inferences for retrieval. More to the point, problematic inferences related to the semantic content of documents do not have to be made. This is really the key idea of the language modeling approach to retrieval. The resulting model is conceptually simple and does not require heuristics to estimate the probabilities.

Most retrieval systems use term frequency, document frequency and document length statistics. Typically these are used to compute a *tf.idf* score with document

length normalization. An example of this is the INQUERY ranking formula shown in Section 4.4.4.

In the approach taken here, collection statistics such as term frequency, document length and document frequency are integral parts of the language model and are not used heuristically as in many other approaches. For this reason, the standard *tf* and *idf* scores will not be used. In addition, length normalization is implicit in the calculation of the probabilities and does not have to be done in an *ad hoc* manner.

Other probabilistic approaches to retrieval are discussed in chapter 2. The definition and study of the language modeling approach appears in Chapter 4 and continues in chapters 5 and 6 which develop additional aspects of the retrieval model. Before that, an additional application of language modeling to a problem in IR, the topic segmentation problem, is introduced.

1.4 Language Models for Topic Segmentation

1.4.1 Motivation

Topic segmentation is potentially useful for information retrieval and related tasks. Early retrieval systems were tested using document collections such as the Cranfield collection [13] which consisted of 1400 relatively short documents. More recently, due to the Text Retrieval Evaluation Conferences (TREC) [25] , much larger collections have been made available. These are heterogeneous collections containing documents from a variety of sources and consisting of several gigabytes of full text documents. Some documents in these collections are several megabytes in size. One problem that arises in this context is that very long documents may be about several different topics. This is a problem for two reasons. First of all, a document with a small relevant section may receive a low ranking due to the 'noise' in the remainder of the document. Secondly, false hits can occur when different query terms appear in unrelated portions of a document.

Modern IR systems do well on these collections in spite of these problems, but the problems will continue to get more difficult as the systems are applied to more and more heterogeneous databases. For example, consider the World Wide Web, where one finds short articles, such as typical USENET news posts, as well as entire books and everything in between. In such an environment, how does one define the concept of document? Perhaps an online book is a document. Then again, maybe a chapter or a chapter section or even a single paragraph should be considered a document. What is the one best way to define the concept of document?

This question may not have a satisfactory answer, but perhaps it does not matter if the problem is redefined. Instead of being concerned with the retrieval of relevant documents, the task can be framed as the retrieval of relevant text segments. In order to make this redefinition, one needs a method of segmenting text by topic.

1.4.2 Examples

Consider an example information need such as labor relations. A query for that information need might contain the words "union" and "management." The first problem is that in large heterogeneous text collections, a document with a small relevant section may receive a low ranking due to the "noise" in the remainder of the document. The *tf* statistic is a function of the number of term occurrences normalized by the size of the document (where size can be measured in various ways). If, for example, an online book about baseball contained a short passage about union/management relations, the terms in this section would be given relatively low weight since they account for a small portion of a very long document. This reflects the intuition that this particular document is not about union/management relations, for the most part. However, the small section about union/management relations by itself may be relevant to the example query and as such one would like to see it reasonably high in the ranked list.

The second problem was that false hits can occur when different query terms appear in unrelated portions of a document containing several topics. For example, in a book review article, two unrelated sections, such as a review of a book about the Union Army during the American Civil War and a second unrelated review of a book about time management could make this article appear relevant to the example query about labor relations especially if the words "union" and "management" occurred frequently enough; it would be preferable to split this article into two documents for the purposes of ranked retrieval.

1.4.3 Text Filtering and Routing

An additional task studied by the IR community is text routing or filtering. In this task, documents are received over a newswire and each document is compared to a collection of user profiles and sent to those users who are likely to be interested. One generally thinks of newswire documents as having predefined boundaries. Topic segmentation does not seem necessary for this task. However, a variant of this task is the routing of speech data from a radio or television broadcast. This data will be indexed by transcripts obtained from an automatic speech recognition system. The output of the speech recognition system is a stream of text. Once again, there is a need to segment the text by topic.

1.4.4 Topic Detection and Tracking (TDT)

The specific task that motivated the segmentation work presented here was the Topic Detection and Tracking (TDT) pilot study [64]. TDT is a study investigating the problems of new event detection and long term event tracking in broadcast speech. Stated briefly, the new event detection task is to label each story as to whether it is the first description of a real world event, the Kobe earthquake for example, that the system has processed. The tracking task is to identify subsequent articles for a topic of interest. For example, if the system identified the Kobe earthquake as a new event,

a user could then decide to track the stories about that event over time. The system would label incoming articles as to whether they describe the Kobe earthquake and the ones that did would be sent to the user.

As mentioned, the data for this task consisted of speech transcripts of broadcast news. Segmentation can be viewed as an enabling technology for this task since story boundaries are not available from automatic speech transcripts.

Over the course of the TDT pilot study [64], which will be described in Section 3.2, the concept of topic was abandoned and an event driven definition was adopted instead. For further discussion of events and their properties see [4]. The operational definition used for the segmentation task in the TDT study was the identification of story boundaries. The original story boundaries were provided for the study corpus by the human transcribers and the task was to recover those boundaries. This definition is somewhat limited in scope, but will be useful in the context of the segmentation study discussed in Chapter 3 and for news routing in general.

1.5 Research Contributions

This thesis makes the following contributions to the field of information retrieval:

- A language modeling approach to the segmentation of text by topic.
- A new approach to text retrieval in general, based on probabilistic language modeling.

1.5.1 Language Models for Text Segmentation

Two complementary feature sets, referred to as discourse features and content features are studied separately for the segmentation problem and then are combined into a single unified probabilistic model. It will be demonstrated that:

1. Text segmentation can be solved in a purely statistical manner without resorting to expensive knowledge-based approaches.

2. Discourse features and content features are useful complementary classes of features.

3. Local Context Analysis (LCA), which has been shown to be effective for solving the vocabulary mismatch problem between queries and documents, can also be used to solve the terseness problem in the context of segmentation.

It will be shown that language models can be used, in lieu of semantic knowledge, for topic segmentation problem. The resulting method is tested in the context of the event tracking task. It will be shown that the segmenter predicts topic boundaries in the text well enough to be almost indistinguishable from the original boundaries with respect to performance of the new event detection task.

The two feature sets, discourse features and content features are first tested separately and it will be shown that both sets of features provide reasonable segmentation performance. However, it will also be shown that when these two feature sets are combined by means of a probabilistic model, the error rate is improved over the methods using either feature set alone.

Finally, the method of local context analysis will be used as a method of identifying content shifts. When segmenting text, words are not always repeated enough times to identify topic shifts entirely by the number of repeated words. It will be shown that the method of local context analysis can be used to improve the accuracy of the content based segmenter. The segmentation phase is detailed in chapter 3.

1.5.2 Language Models for Information Retrieval

The new retrieval model is developed in chapters 4 though 6. This model is tested using Labrador, a research prototype retrieval system described in section 4.4.3, and compared to a method of retrieval based on more traditional models of retrieval. It will be shown that:

1. The retrieval problem can be simplified by phrasing it as a generative language modeling problem and the resulting approach will provide effective retrieval.

2. Heuristic techniques used by previous retrieval models are not necessary to provide effective retrieval. A purely probabilistic method can be used instead.

3. Additional techniques, such as relevance feedback, which often involve heuristic estimation in existing models, can be derived from the language modeling approach without the use of heuristics.

Standard probabilistic models of retrieval have viewed the problem as an inference process, whereby the probability of relevance can be estimated from the presence of index terms in a document. The process of determining these index terms is also one of probabilistic inference. The language modeling approach is conceptually simpler than existing models of retrieval. Collection statistics, used heuristically in existing retrieval models, can be used directly in the estimation of probabilities in the language modeling approach. Furthermore it will be shown that the process of relevance feedback which in current models relies heavily on heuristic use of collection statistics can be derived directly from the language modeling approach.

A description of additional aspects of retrieval will also be provided to show that other techniques used by retrieval systems have a natural interpretation in the language modeling approach.

CHAPTER 2

RELATED WORK AND BACKGROUND MATERIAL

This chapter introduces previous work related to language modeling, information retrieval and text segmentation. The chapter begins with a discussion of language models and probability estimation. Several of the language modeling techniques described later will make use of some form of smoothing and so an introduction to smoothing methods is included. After that, various language modeling techniques and some of their applications are discussed. Language modeling is the central concept for the solution of the problems addressed in this thesis and so the work described in this section provides important background for the understanding of the whole approach. In Section 2.1.8, language modeling approaches to Asian language word segmentation are described. This is included here because it is related to work discussed in Chapter 7.

The chapter continues with discussion of other models of information retrieval. This is important background for the discussion of the new approach to retrieval discussed in chapter 4. This section includes a discussion of passage retrieval. It will be shown in Section 6.2.4 that passage retrieval has a natural interpretation in the language modeling approach to retrieval investigated in this thesis. The two next sections, 2.2.8 and 2.2.7 describe query expansion techniques and relevance feedback, respectively. These are two techniques for improving retrieval effectiveness. Both of these techniques will be shown to have a natural interpretation in the language modeling approach to retrieval in Sections 5.2 and 5.3.

Section 2.3 describes other approaches to text segmentation, the application of language modeling studied in Chapter 3.

2.1 Language Models

2.1.1 Smoothing Methods

The topic of smoothing encompasses a wide variety of techniques that can be applied to probability estimation and many related problems. The discussion here will be confined to the questions of why smoothing is useful, when it should be done, and an example of a simple smoothing technique to make these notions more concrete. Additional smoothing methods will be introduced later in this chapter in the context of specific probability models.

Consider the problem of estimating the probability that a real valued random variable X takes on the value x. Assume that a limited amount of training data is available from which to estimate this probability and we wish to perform the estimation by counting. Because X is a a real-valued variable, by definition there is not enough data to estimate the entire distribution. Even in the discrete case, it is often very difficult to get enough data to estimate a distribution purely from counts. A useful notion in discussing the errors made by estimators derived from training data is known as the bias-variance tradeoff.

Stated simply, bias error is error due to the inability of an estimator to fit the data closely enough. On the other hand, variance error is error due to the ability of an estimator to fit the data too well, also known as over-fitting.

For example, in Figure 2.1 data was generated from a logistic function corrupted by Gaussian noise. Suppose these data are used to build an estimator for novel data from the same source. In other words, for a given value on the x axis, the task is to predict the corresponding y value. Consider two different estimators.

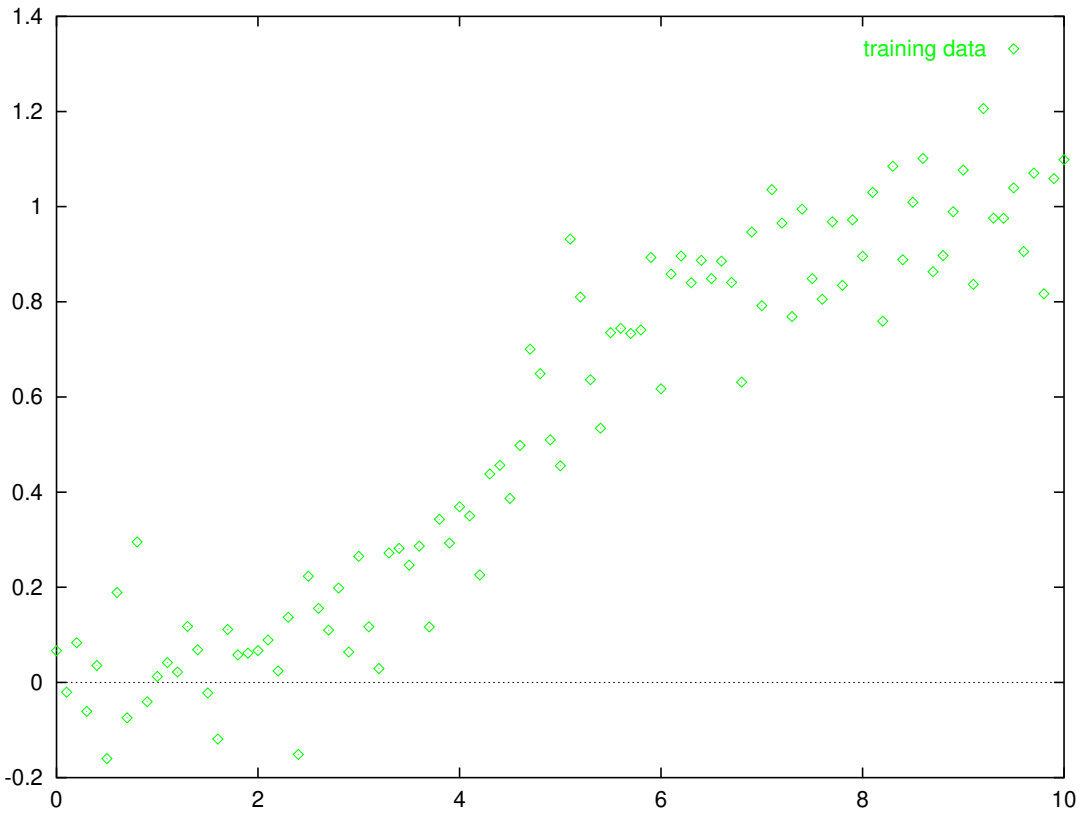

Figure 2.1. Example training data from a logistic function corrupted by Gaussian noise.

The first estimator is obtained by connecting the original training data points. The resulting curve is shown in figure 2.2, superimposed on a second set of data points from same source and, again, perturbed by random Gaussian noise. Notice that there is considerable error in the prediction made by this estimator since it has, in effect, fit the noise in the training data as much as it has fit the underlying function. This is error due to variance, i.e., error as a result of fitting irrelevant artifacts of the data.

The second estimator is a linear fit to the training data. The resulting curve is shown in figure 2.3. Notice that, once again, there is considerable error in the prediction made by the estimator. However, in this case, the error is more systematic than in the previous example. In other words, the linear fit tends to systematically over or under estimate the y value for large regions of the graph. Recall that the data

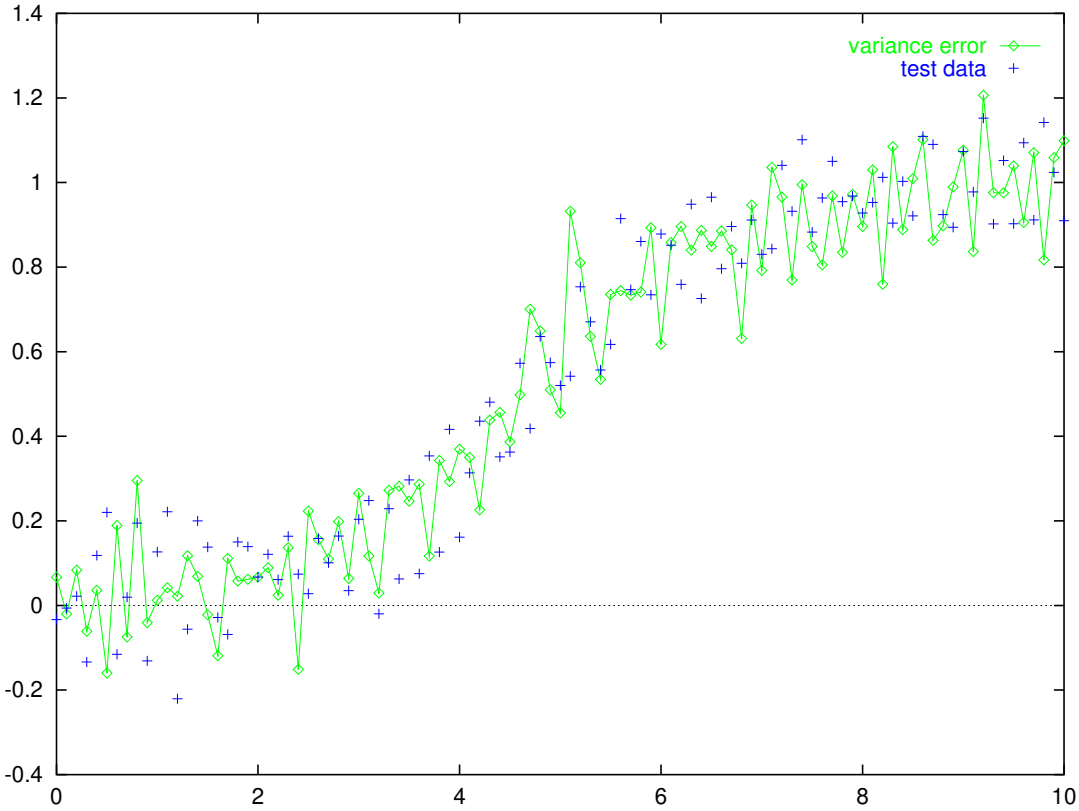

Figure 2.2. Example of an estimator that exhibits variance error.

was generated by a logistic function. The linear constraint does not allow this estimator to fit the underlying form of the data very well. In cases where the generation mechanism is unknown, choosing the correct form of an estimator can be crucial for avoiding excessive bias error.

How does the bias variance trade off relate to density estimation, the prediction of the probability that a random variable X takes on the value x? As mentioned, there is, more often than not, insufficient training data from which to estimate a probability density function purely by counting. What does one do instead? One possibility is to assume a parametric distribution and use the training data to estimate the parameters. For example, one can assume a normal distribution and estimate the mean and variance. This allows all of the data to be used to estimate two values and is a very good solution to the problem if the underlying data really is

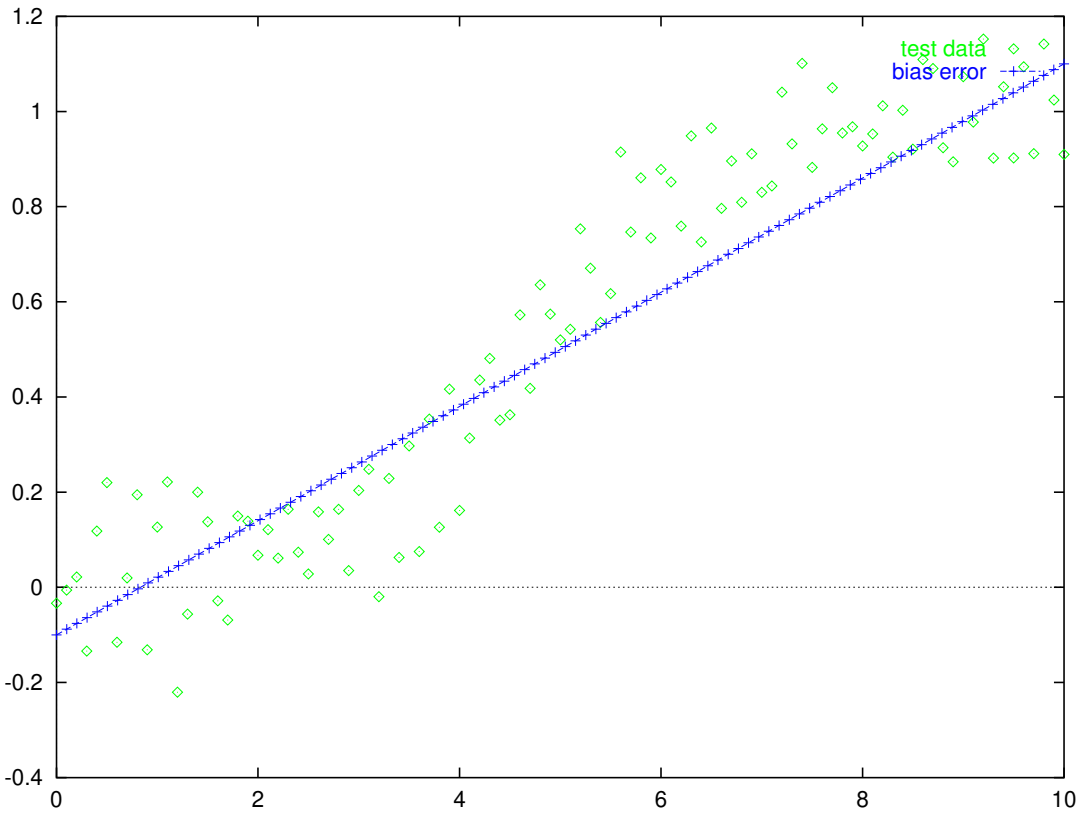

Figure 2.3. Example of an estimator that exhibits bias error.

normally distributed (or well approximated by a normal distribution in a region of interest). However, if the data is not normally distributed, making this assumption will introduce bias error. If the data is far from normal, this error may be quite severe.

On the other hand, if one estimates the probability purely from the counts, variance error will be introduced. One common symptom of variance error in language modeling is that an unseen word is assigned a probability of zero. The resulting model performs badly on novel data as a result.

In between these two extremes are non-parametric density estimators. The bias-variance tradeoff is still present in non-parametric estimation but these two sources of error can be balanced to obtain a reasonable estimator. A simple example of a non-parametric estimator is the histogram.

Histogram estimators are probably the simplest form of non-parametric density estimator or smoothing-based estimator. The fixed-width histogram has a single smoothing parameter known as the bin width, h. Getting back to the original problem of estimating $P(X = x)$, the X values are divided into bins of width h. Then, each training data point is put into the appropriate bin. The density estimate for a data point is then computed as:

$$\hat{P}(X = x_j) = \frac{\#x_i}{nh}$$

where $\#x_i$ is the count of points in the same bin as x_j, n is the number of data points and h is the bin width. This value is used to estimate the density of any future data point that would be assigned to the bin.

As bin width is increased, bias error is increased and variance error is reduced and, as expected, the converse is also true. Intuitively, this is the case because, as bin width approaches zero, the estimator becomes the simple count estimator, which fits the data exactly, random noise included. This can cause error due to variance. As the bin width gets wider, there are fewer distinctions between points and, in the extreme, all of the points would be placed in a single bin and the average value would be used to estimate everything, which can cause error due to bias.

For additional material on smoothing methods, see [57]. Specific examples of smoothing will be presented in the context of various language modeling problems later in this chapter.

2.1.2 Predictive Language Modeling

In general, language models are probability distributions over strings in a given language. The language itself can be any set of sequences made up from a predefined set of characters. In the context of information retrieval, the sequential aspects of language do not need to be modeled in the case of single word features and so, for

document retrieval, the specific language models in which we shall be interested are those that predict the probability of word tokens without respect to order. This can be thought of as "bag of words" modeling rather than string modeling. Conversely, for the topic boundary prediction problem, sequence is important, but only at a gross level. However, language models are used in a variety of tasks where fine grained sequential aspects of the data are modeled and we shall consider some of them here.

2.1.3 Language Models for Speech Recognition

Bahl *et al.* [6] describe the use of of language modeling to improve the performance of speech recognition systems. This is especially important to reduce error for large vocabulary recognition. The speech recognition task can be framed as the maximum likelihood estimate of a series of words given an acoustic signal. One could, in principle, solve this problem by means of an acoustic model for each word in the vocabulary. However, doing so would require search in a very large space of possible acoustic models and would be prone to error. Language modeling is used to narrow the search by using the predictive probabilities as a prior over the possible acoustic models. If one can predict words that are more likely to occur at a given point in the speech signal, the search space becomes narrower and error can thereby be reduced.

A method of modeling that works well for the purpose of word prediction is the simple trigram model. The probability of a word w at a given position in the text is estimated by the conditional probability $p(w|w_{-1}w_{-2})$. In other words, the Markov assumption is being made with the states consisting of the previous two words. Estimation of the trigram probabilities is done by means of maximum likelihood estimation from a large training corpus. Since the possible number of trigrams is cubic in the vocabulary size, many possible trigrams may not occur in a given training collection. This sparse data problem will be addressed using a type of smoothing known as a backoff model. In general, backoff models are methods where the probabilities

of relatively rare events are estimated using data from more common events. Specifically, to estimate trigram probabilities, bigrams and unigrams will also be used. Since bigrams and unigrams are more common than trigrams, more data exists for the estimation of their probabilities. For trigrams with too few occurrences, the bigram is used. Likewise, for rare bigrams, unigram probability estimates will be used.

The specific form of the estimator is a linear interpolation model of the following form:

$$\hat{p}(w|w_{-1}w_{-2}) = \lambda_3\hat{p}_{ml}(w|w_{-1}w_{-2}) + \lambda_2\hat{p}_{ml}(w|w_{-1}) + \lambda_1\hat{p}_{ml}(w) + \lambda_0 1.0/|vocabularysize|$$

where $\hat{p}_{ml}(\cdot)$ is estimated from the frequencies in the training corpus and the interpolation parameters λ_i are estimated using a method known as deleted interpolation, described below.

Typically, several λ vectors will be estimated for trigrams of different frequencies. For low frequency trigrams, more weight would be given to the bigrams and so forth. The multiple vector models are a form of generalized backoff model where one "backs off" to the bigram estimate when there is not enough data for a particular trigram and likewise one "backs off" to the unigram probability when there is insufficient data to estimate the bigram etc. The implementation of this scheme requires the definition of several ranges of trigram frequencies and the estimation of λ vectors for each range [6]. For example, one may wish to define three ranges of frequencies for common, medium frequency, and rare trigrams, and to estimate a set of parameters for each one.

In order to estimate the interpolation parameters, the data is divided into equally sized blocks. To estimate the λ values for a particular frequency range, the n-grams of that range are counted in all blocks except one. The held out block is used to estimate the number of unseen n-grams. Each block is held out in turn in order to make the best use of the available training data.

As mentioned, in the context of information retrieval, one is not, in general, interested in predicting the next token in a sequence and so the language modeling that will be employed in Chapter 4 will be designed to estimate the probability of a bag of words and local predictive effects will not be modeled. However, a simple backoff scheme will be employed for non-occurring query terms. An estimate of the collection-wide occurrence probability will be used to estimate the probability of words not occurring in a document of interest. This will be described in detail in Chapter 4. Likewise, for the segmentation task, a backoff method will be used to handle unseen words. For the segmentation task, sequential modeling *is* used, but at a coarser level than the trigram models just described.

2.1.4 Exponential Models

A richer class of predictive models based on the generalized Gibbs distribution has been explored by Darroch and Ratcliff [16] and later by Della Pietra *et al.* [18]. This class of models can account for longer range effects than the simple trigram model.

The distribution is defined as follows:

$$\hat{p}(x) = \frac{e^{\lambda \cdot f(x)} p_d(x)}{\sum_X e^{\lambda \cdot f(x)} p_d(x)}$$

where λ is a vector of parameters, $f(x)$ is a vector of features that predict x and $p_d(x)$ is a default probability estimate of x. For example, for a predictive language modeling task, x would be a word, $p_d(x)$ could be a backoff trigram model, as defined in Section 2.1.3, and $f(x)$ could be a set of long range predictors of x. For example, the fact that x has occurred previously will tend to make it more likely to occur again. That being the case, one of the features in the vector $f(\cdot)$ might be a function that evaluates to one if x has occurred in the last 300 words and to zero otherwise. The parameters of the model can be learned using a variety of algorithms including

Generalized Iterative Scaling, developed by Darroch and Ratcliff [16] and Improved Iterative Scaling algorithm developed by Della Pietra et al. [18].

Beeferman *et al.* [8] applied this class of models to the text segmentation problem. See Section 2.3 for a description of this work and of the specific implementation of the estimation algorithms.

2.1.5 Hidden Markov Models (HMMs)

Hidden Markov Models are probabilistic finite automata. To begin with, finite automata can be defined in two ways, the Mealy machine and the Moore machine. In the Mealy formulation, the symbols are associated with the transitions while in the Moore formulation, symbols are associated with the states. In Figure 2.4, an example state diagram is shown for a Mealy machine that writes 'A' zero or more times followed by one or more occurrences of 'B.' The state on the left is the start state, indicated by the arrow from the left, and the state on the right is the end state indicated by the double circle. Note that the symbol is written during the transition from one state to the next. Figure 2.5 shows a Moore machine for the same language. In this case, the symbol is written during the occupation of the state rather than during the transition.

In most applications, Hidden Markov Models will defined as Moore machines. In other words, the output symbols will be associated with the states. The reason for this will be made clearer with an example application, part-of-speech tagging, described in Section 2.1.6, but first the models will be defined in more detail.

A second distinction to be made between finite automata is the deterministic machine vs. the non-deterministic machine. The two machines in figures 2.4 and 2.5 are both deterministic machines. Non-deterministic equivalent machines are shown in figures 2.6 and 2.7. Notice that in these machines, the output of the machine does

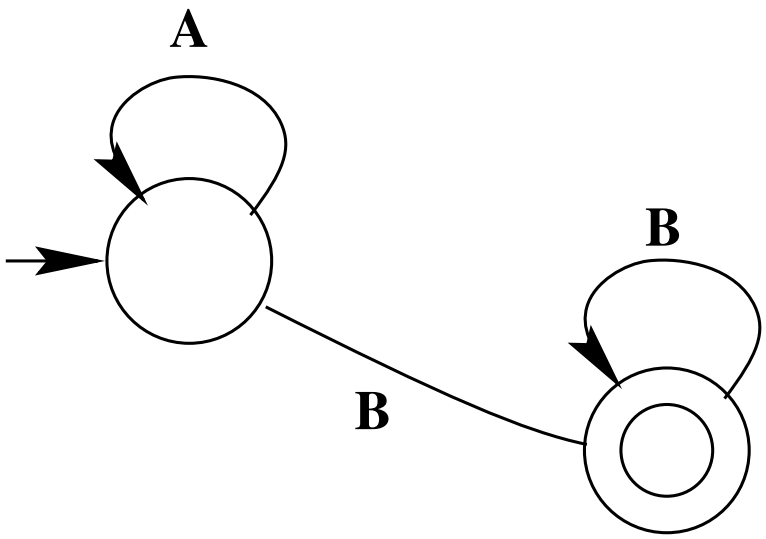

Figure 2.4. Example of a Mealy machine.

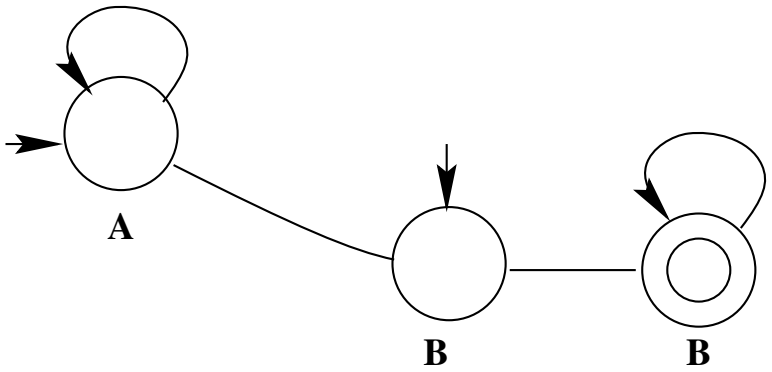

Figure 2.5. Example of a Moore machine.

not determine the state. For example, in Figure 2.6, the machine can write a 'B' and make the transition to either of the two right hand states.

Probabilistic automata can be thought of as a generalization of non-deterministic automata. As stated earlier, Hidden Markov Models are probabilistic finite automata. Consider the case of the non-deterministic automaton. If one views the possible non-deterministic steps as equally likely to occur, then the non-deterministic machine can be viewed as a special case of the probabilistic machine. The HMM generalizes the non-deterministic finite automaton in that any probability distribution can be placed

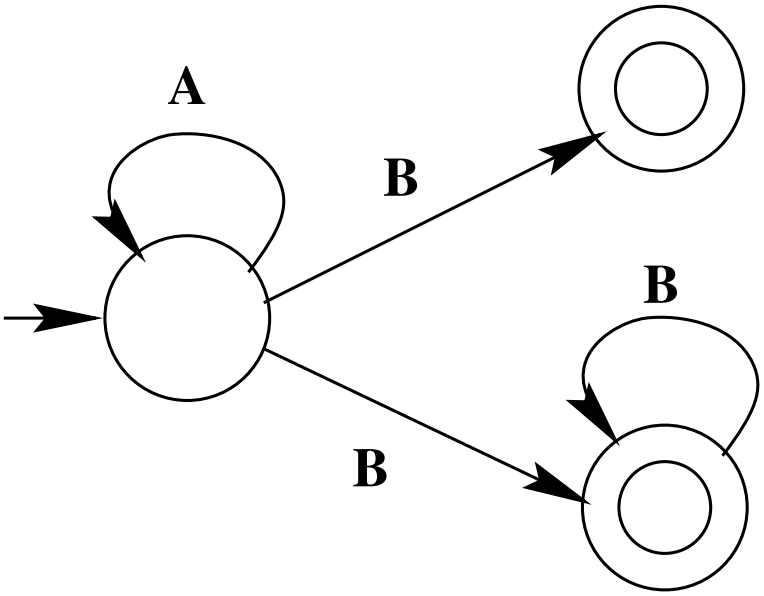

Figure 2.6. Example of a non-deterministic Mealy machine.

over the transitions. In addition, several output symbols can be associated with a state along with a probability distribution over the possible outputs.

The parameters of an HMM consist of a vector π and two matrices A and B. π is the initial state probability distribution of the model [44].

$$\pi_i = \text{the probability of starting in state } i, 1 \leq i \leq |states|$$

The matrix A is the transition probability matrix,

$$A_{ij} = \text{the probability of state } j \text{ at time } t+1 \text{ given state } i \text{ at time } t$$

The matrix B is the probability distribution of the output symbols.

$$B_{ij} = \text{the probability of writing symbol } j \text{ given that the model is in state } i$$

Intuitively, the underlying state space is hidden and can only be observed indirectly via the sequence of observed symbols. For this reason, HMMs are also referred

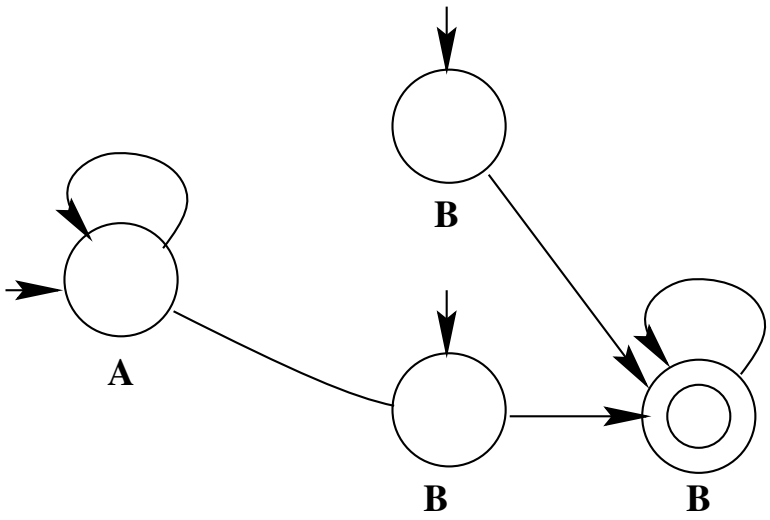

Figure 2.7. Example of a non-deterministic Moore machine.

to as Markov source models. In other words, the underlying source is a Markov chain but the observable sequence has been corrupted by noise.

Computational problems associated with HMMs can be divided into three problems as was done by Rabiner [44]. The treatment of HMMs given here is largely derived from the discussion in Rabiner [44] and Rabiner and Juang [45]. For an alternative formulation of HMMs see Elliot, Aggoun and Moore [21].

The three problems discussed by Rabiner are:

1. Calculating the probability of a sequence given the model.

2. Estimating the model parameters.

3. Estimating the maximum likelihood state sequence for an observation.

In the discussion that follows, it will be assumed that a sequence of symbols has been written by a specific Hidden Markov Model. The computational problem associated with the first item is that for a sequence of length t and an HMM with n states, there are n^t possible paths through the model, i.e. there are n^t possible sequences of states. However, due to the Markov property, the probability of being in

a state n_i at time t_j depends only on the state at time t_{j-1}. This property allows for efficient calculation of the probability in time $O(tn^2)$ using dynamic programming.

The second problem is that of estimating the model parameters, the A and B matrices and the vector π. Several methods exist for estimating these probabilities and some of them will be described here.

Finally, the third problem is that of determining the maximum likelihood state sequence for a given output sequence of symbols. This problem is often referred to as the decoding problem. The same computational difficulty that arose in the probability calculation, an exponential number of paths, is also an issue for the decoding problem. Once again the answer to this problem will be dynamic programming, in this case in the guise of the Viterbi algorithm, to be discussed.

As pointed out earlier, a serious computational problem must be solved in order to calculate the probability that a sequence of symbols would be written by a specific Hidden Markov Model. The naive computation of the probability is to sum over all possible state sequences. To make this more concrete the following notation will be used:

M	the model , a fully specified HMM
n	the number of states of the model M
t	the length of the sequence
O_i	the sequence of symbols where i ranges from 1 to t
s_i	the state at time i where i ranges from 1 to t
π_k	the probability of beginning a sequence in state k
$A_{k,k'}$	the probability of making a transition from state k to state k'
$B_{k,l}$	the probability of writing symbol l while in state k

Using this notation, the probability of the observation sequence O for a single pre-specified state sequence is written as follows:

$$\pi_{s_1} B_{s_1,O_1} A_{s_1,s_2} B_{s_2,O_2} \cdots A_{s_{t-1},s_t} B_{s_t,O_t}$$

Intuitively, this probability is defined as the probability of starting in the initial state and writing the first character, making the transition to the second state and, from there, writing the second character, etc.

Given an estimate for the probability of the sequence given a specific state sequence one can calculate $p(O|M)$, the probability of the sequence given the model by summing over all possible state sequences. However, in the general case, all of the entries of A and B can be non-zero meaning that, at each timestep, the model can make a transition to any of the n states and write the observed output symbol. This means that the total number of state sequences is exponential in t, the length of the symbol sequence.

In order to avoid summing over an exponential number of sequences, a dynamic programming method, commonly known as the forward algorithm can be used. The forward algorithm takes advantage of the Markov property by keeping an $n \times t$ table of probabilities for each state at each time step. This forward probability table will be referred to as $\alpha_{i,k}$, where i is the index of time and k is a state. Since the Markov property says that the probability of being in a state k at time i is conditioned only on the state at time $i-1$ (or, in general, $i-\delta$ for higher order processes), the forward probability table can be calculated as follows:

$$\begin{aligned} \alpha_{1,k} &= \pi_k B_{k,O_1} \\ \alpha_{i+1,k'} &= \left[\sum_{k=1}^{n} \alpha_{i,k} A_{k,k'}\right] B_{k',O_{i+1}} \end{aligned}$$

In other words, the probabilities are propagated forward through the table and the current timestep is calculated from the current output symbol, the state table

entries from the previous time step and the transition probabilities from each of the previous states. The probability of the symbol sequence is then the sum of the final column of the table.

One can also define a backward variable β where a similar calculation is done from the end of the sequence to the beginning. The sum of the first column of the backward table yields the same probability estimate as the corresponding forward calculation. Both the forward and backward variables are used for the estimation of the model parameters described next.

Several methods exist for the estimation of HMM parameters. The two methods discussed here include estimation from labeled training data and the Baum-Welch algorithm for the estimation from unlabeled training data.

The easiest method for estimating HMM parameters is to hand-label a large quantity of data with state information. The parameter estimation is then a simple matter of counting the number of state transitions and the number of times each symbol is labeled from each state and estimate the probabilities from the counts. Assuming that the amount of hand labeled data is not enough to achieve reliable estimates, one may wish to do some smoothing.

For example, suppose an HMM is used to model the production of natural language text. Each word in the text can be labeled with a state tag and the probabilities can then be estimated from the counts. However, suppose a word w is not labeled with a state s anywhere in the training data. The probability of w given s in the B matrix would be assigned the value zero based on the count information. In many cases, this is not a reasonable value since it amounts to assuming that an event is impossible only because it has not been seen in a limited amount of training data. In order to avoid this unreasonable assumption one can assign a small additional 'mass' to the output distribution of s and distribute the unseen words over that mass according to their global probability of occurrence. The column vector associated with state

s can then be renormalized to insure that the state has a valid output probability distribution.

The Baum-Welch algorithm utilizes the forward and backward variables, defined earlier, in order to estimate the model parameters from unlabeled training data. The model parameters can be initialized randomly or according to a reasonable prior and then these initial values are updated over several iterations of the algorithm. As a variant of the Expectation-Maximization (EM) algorithm, the Baum-Welch algorithm proceeds by calculating the expected number of transitions made from each state and the expected number of transitions between states, both conditioned on the current parameter values, and the observation sequence (i.e., the training data).

How can these expectations be calculated? The calculation is done by first defining two new variables:

$$\begin{aligned} \gamma_{i,k} &= p(s_i = k|O, M) \\ \zeta_{i,k,k'} &= p(s_i = k \wedge s_{i+1} = k'|O, M) \end{aligned}$$

In other words, γ is the probability that the state at time i is k given the training data and the current parameter values, while ζ is the probability of making a transition from state k to state k' at time i given the training data and the current parameter values. These variables can be calculated from the forward and backward tables as follows:

$$\begin{aligned} \gamma_{i,k} &= \frac{\alpha_{i,k}\beta_{i,k}}{p(O|M)} \\ \zeta_{i,k,k'} &= \frac{\alpha_{i,k}A_{k,k'}B_{k',O_{i+1}}\beta_{i+1,k'}}{p(O|M)} \end{aligned}$$

So, $\gamma_{i,k}$ is calculated from the forward and backward estimates normalized by the probability of the training sequence given the current model parameters. Likewise, $\zeta_{i,k,k'}$, the probability of making a transition for state k to state k' at time i is calculated as the probability of getting to state k at time i ($\alpha_{i,k}$) making the transition to

state k' ($A_{k,k'}$), writing the corresponding symbol from state k' ($B_{k',O_{i+1}}$) and then seeing the remainder of the sequence starting from state k' ($\beta_{i+1,k'}$) and, again, normalized by the probability of the training data given the current model parameters.

The desired expectations can then be calculated by summing over t, the length of the training sequence, as follows:

$$\begin{aligned}
\textstyle\sum_{i=1}^{t} \gamma_{i,k} &= \text{expected transitions from } k \\
\textstyle\sum_{i=1}^{t} \zeta_{i,k,k'} &= \text{expected transitions from } k \text{ to } k'
\end{aligned}$$

This concludes the expectation step of the EM algorithm. The updates of the model parameters are then completed as follows:

$$\begin{aligned}
\hat{\pi}_k &= \gamma_{1,k} \\
\hat{A}_{k,k'} &= \frac{\sum_{i=1}^{t} \zeta_{i,k,k'}}{\sum_{i=1}^{t} \gamma_{i,k}} \\
\hat{B}_{k,l} &= \frac{\sum_{i=1 \wedge O_i=l}^{t} \gamma_{i,k}}{\sum_{i=1}^{t} \gamma_{i,k}}
\end{aligned}$$

In other words, $\hat{\pi}_k$, the start probability for state k, is estimated as the probability of making a transition from state k at time 1. Similarly, $\hat{A}_{k,k'}$, the transition probability from state k to state k' is estimated as expected number of transitions from state k to state k' divided by the expected number of transitions from state k. Finally, $\hat{B}_{k,l}$, the probability of writing symbol l in state k is estimated as the expected number of times of being in state k and writing symbol l divided by the expected number of times in state k. This procedure is applied repeatedly until convergence.

Finally, the third problem associated with HMMs is the estimation of the maximum likelihood state sequence for an observation. Once again, the exponential number of possibilities must be contended with and, as before, the solution will be a dynamic programming algorithm. The Viterbi algorithm [63] is similar to the forward algorithm for determining the probability of a sequence. However, in this case,

the probability of the sequence following the maximum likelihood state sequence is calculated instead. Recall that the forward calculation proceeds as follows:

$$\begin{aligned} \alpha_{1,k} &= \pi_k B_{k,O_1} \\ \alpha_{i+1,k'} &= \left[\sum_{k=1}^{n} \alpha_{i,k} A_{k,k'}\right] B_{k',O_{i+1}} \end{aligned}$$

Where each entry in the α table is updated by summing over the entries from the previous time step. A similar procedure is used in the Viterbi algorithm, but the table δ will be filled in using the maximum at each step as follows:

$$\begin{aligned} \delta_{1,k} &= \pi_k B_{k,O_1} \\ \delta_{i+1,k'} &= \left[\max_{1 \le k \le n} \delta_{i,k} A_{k,k'}\right] B_{k',O_{i+1}} \end{aligned}$$

At each time step, a backtracking table Ψ is also filled in:

$$\Psi_{i+1,k'} = \arg\max_{1 \le k \le n} \delta_{i,k} A_{k,k'}$$

The Ψ table is filled in with the most likely previous state, k, that was on the path preceding state k'. When the tables have been filled in, the maximum likelihood path is computed by finding the maximum value in the final column of the δ table and then backtracking through the Ψ table to determine the sequence of states.

2.1.6 Example: POS Tagging

An example of HMMs in practice is the Xerox part-of-speech tagger developed by Kupiec [36]. The underlying state space consists of the part-of-speech tags. In this context the transition matrix A, introduced in Section 2.1.5, consists of the matrix of conditional probabilities of seeing a particular tag at the current time step given the tag seen at the previous time step.

In each state, the output symbols are all words in the vocabulary. A special unknown word symbol is included for words that did not occur in the training data.

That means the output matrix B is a matrix of conditional probabilities of each word given the current tag. Finally, π is a vector of probabilities of starting a sentence with each of the tags. As mentioned, this formulation is known as a Moore machine where the output symbols correspond to the states. This is a very natural formulation of the problem because the task is seen as uncovering the hidden state sequence given the observable sequence of outputs.

The model probabilities are estimated via the Baum-Welch algorithm, the Expectation-Maximization algorithm for HMMs described previously. The process of tagging now becomes the process of determining the maximum likelihood state sequence for a given sequence of words, i.e., the sequence of tags that would produce the text with maximum likelihood given the HMM parameters. This is accomplished using the Viterbi algorithm, the dynamic programming algorithm for determining the maximum likelihood path through the model that was described in Section 2.1.5.

2.1.7 Acoustic Models for Speech Recognition

In addition to the trigram language models described earlier, the process of speech recognition makes use of HMMs for acoustic modeling of the speech signal. For this task, each word has an acoustic model where the state space is the underlying sequence of sounds making up the word and the output symbols account for the variations in pronunciation for each sound.

In Section 1.2, the concepts of "phone" and "phoneme" were introduced. A phone is the actual sound produced, while a phoneme is an abstract underlying sound which is essentially an equivalence class of various phones. In terms of the definition of HMMs given in Section 2.1.5, the A matrix represents the transitions from one phoneme to the next. The B matrix represents the probability distribution over possible phones for a particular phoneme.

The speech signal is first quantized into a sequence of discrete observations and the probability of the given word is computed according to the acoustic model and also according to the trigram language model described in Section 2.1.3. The two estimates together are used to model the joint distribution of the word and its acoustic signal [6].

This concludes the introductory discussion of probabilistic language models. Additional material on language models will be covered in the context of specific applications in sections 2.3.3 and 2.3.4.

2.1.8 Asian Language Word Segmentation

An additional application of language modeling to information retrieval arises in the context of retrieval of Asian language documents. Asian languages such as Chinese, Japanese, Thai, Korean, and many others, are written without interword delimiters. Retrieval systems generally rely on some form of word segmentation as part of the tokenization process. For an overview of Chinese segmentation in the context of information retrieval see [66]. The task of segmentation is often carried out by means of probabilistic language models.

Ponte and Croft [43] did a study in which they developed a retargetable segmenter for Asian languages. The core of the segmenter was a probabilistic automaton, a Markov model, of the lexicon. This method was shown to be accurate, easily retargetable and fast enough to index large text collections.

Sproat *et al.* [58] describe a Chinese word segmentation algorithm based on probabilistic automata. The details of the probability models were not discussed but it was mentioned that their approach included special recognizers for Chinese names and transliterated foreign names and a component for morphologically derived words.

Barnett [7] used a word based model for segmenting Japanese. The model used word frequency. The input is segmented by summing the frequencies of words in each

possible segmentation and subtracting out a per word cost. Additional complexity arises in Japanese from inflectional endings. In order to consider a candidate word, morphological processing is done to find every possible grouping of characters that form valid words.

The topic segmentation methods that will be described in chapter 3, will be somewhat similar in nature to the word segmentation discussed here. The main difference is that topic segmentation has a hidden state that will be modeled by a Hidden Markov Model whereas the word segmentation algorithms can be described as observable Markov chain models. An interesting question about word segmentation, and many other natural language processing techniques, is that, often, doing a better job of language processing does not necessarily produce better retrieval results. The probabilistic techniques for segmentation discussed here in conjunction with the language modeling approach to retrieval may help provide new ways for investigating this issue. This will be discussed further in section 7.2.4.2.

2.2 Retrieval Models

As noted in Section 1.1.1, there are two widely used classes of retrieval models. One of the two is the vector space model of Gerard Salton [51], discussed in section 2.2.1. Other approaches to retrieval fall under the heading of probabilistic models. These include early models such as the Robertson and Sparck Jones model [46] and the Croft and Harper model [15] as well as more recent probabilistic approaches such as the Fuhr model [22] and the INQUERY inference network model [61].

2.2.1 The Vector Space Model

The sense of the word "model" in the "vector space model" is the sense used in much of computer science. It refers to a useful abstraction of the task at hand. Specifically, for retrieval, one can implement a system using a variety of data struc-

tures depending on performance requirements and on the hardware on which the system runs. The vector space model provides an abstract way of talking about the retrieval process that does not depend on how a specific system is implemented or on the formula used to rank the documents. The model states that documents and queries can be viewed as vectors in a T dimensional space where T is the number of terms in the collection. The retrieval process is then a ranking of documents by the "distance" from the query. This distance can be computed in a variety of ways, cosine correlation being a standard measure, though the model does not specify how to do the distance calculation [51].

As an abstraction of the retrieval process, the vector space model works quite well. It describes the process of retrieval in a way that abstracts away all of the details of implementation and of the specific ranking formula and does so in an intuitive way. However, since the model does not prescribe how the ranking is to be done, it does not provide the researcher with guidance for improving the process.

2.2.2 Probabilistic Models

Models that do attempt to provide this guidance generally fall under the heading of probabilistic models. Here the word "model" is intended to mean probability model. Probabilistic retrieval is usually phrased as the process of ranking documents by estimated probability of relevance to the information need of the user as expressed in a query. Where the vector space model attempts to describe the retrieval process, probabilistic approaches attempt to explain it.

Two well known probabilistic approaches to retrieval are the Robertson and Sparck Jones model [47] and the Croft and Harper model [15]. Both of these models estimate the probability of relevance of each document to the query.

The Robertson and Spark-Jones model weights terms according to the following log ratio:

$$\hat{w}_t = \log\left(\frac{p(t|r) \times (1.0 - p(t|\overline{r})}{(1.0 - p(t|r)) \times p(t|\overline{r})}\right)$$

where $p(t|r)$ is the probability of term t occuring in a relevant document and $p(t|\overline{r})$ is the probability of term t occuring in a non-relevant document. This model depends on relevance information for the estimation of the two probabilities [46].

The Croft and Harper model [15] uses an equivalent ratio for term weighting stated in a different form:

$$\hat{w}_t = \log\left(\frac{p(t|r)}{(1.0 - p(t|r))}\right) - \log\left(\frac{p(t|\overline{r})}{(1.0 - p(t|\overline{r}))}\right)$$

The main difference is that Croft and Harper make two additional assumptions. First, in the absence of relevance information, assume that query terms are equally likely to occur in relevant documents. Second, the probability of occurrence in non-relevant documents can be approximated by $\frac{df_t}{\text{collection size}}$. In other words, because most documents in the collection are not relevant, the non-relevance probability is closely approximated by the collection statistics. This means that relevance information is not needed in the Croft and Harper model [15].

What these two models have in common is that the within document frequency of query terms is not part of the model. This is in contrast to later approaches to probabilistic retrieval, including the approach developed in this thesis. An additional difference between these early models and the new approach developed here is that, in the new approach, relevance is not directly modeled.

Regarding within document frequency of terms, this information is useful for retrieval, as has been shown empirically. A mechanism for incorporating term frequency information is the indexing model which was discussed at length in Chapter 1. More

recent approaches to probabilistic retrieval have attempted to integrate document indexing into the retrieval model as whole.

2.2.2.1 The Fuhr Model

A recent probabilistic model is that of Fuhr [22]. A notable feature of the Fuhr model is the integration of indexing and retrieval models. A major difference between Fuhr's approach and the approach investigated here is that in the Fuhr model, the collection statistics are used in a heuristic fashion in order to estimate the probabilities of assigning concepts to documents. The notion of estimating the probability that a concept should be assigned to a document by means of an indexing term is a problematic one. How can one claim to be able to reliably estimate this quantity? A human indexer will assign terms based on an understanding of the semantics of the document. It would not be surprising if the tendency of human indexers to assign terms was somewhat correlated with the number of term occurrences, but it would be quite surprising if one could obtain an accurate probability estimate of this tendency from the available data. In the language approach described in Chapter 4, heuristic methods of probability estimation are not necessary since there is no inference of concepts from terms and, in fact, no notion of the assignment of indexing terms.

Like the 2-Poisson model, the Fuhr model was developed by analogy to manual indexing. The approach to indexing involves learning of term assignment probabilities from available relevance judgments over the long term. The learning approach is to assume a model and estimate weights using regression. Originally linear regression was used to estimate the models. Later, logistic regression was used partially in response to objections raised by Cooper.

Cooper has criticized Fuhr's use of linear regression to fit relevance judgments because relevance is assumed to be a binary valued random variable [14]. This is an important point as Cooper recommends logistic regression to fit the model to

relevance judgments. Even if one makes the assumption of two valued relevance judgments, using logistic regression to fit the data requires an additional assumption. The assumption is that feature assignment probabilities can be learned for the sets of relevant and non-relevant documents from the available information. One potential problem with this notion is that it is possible for documents to be relevant in more than one way and likewise documents can be non-relevant in more than one way. For example, for the information need "sanctions against Iraq", a document can fail to be relevant if it describes sanctions against Iran instead of Iraq, but it can also fail to be relevant if it doesn't describe sanctions at all. There is a difference between non-relevant documents that are similar to relevant documents (and there a several ways to be similar) and documents that are non-relevant and extremely dissimilar. Fitting a two valued relevance model does not capture this difference.

2.2.2.2 The Inference Network Model

Another recent probabilistic approach is the INQUERY inference network model of Turtle and Croft [60]. Similar to the Fuhr model, Turtle and Croft integrate indexing and retrieval by making inferences of concepts from features. Features include words, phrases and more complex structured features. Evidence from multiple feature sets and multiple queries can be combined by means of a Bayesian network in order to infer the probability that the information need of the user has been met. This distinction between information need and query is a notable feature of this model. As previously noted, in this thesis the emphasis has been shifted from probability of relevance to probability of query production. It is likely that these probabilities are correlated but no attempt is currently made to model that correlation explicitly. This point is discussed further in section 7.2.

Figure 2.8 shows a pictorial representation of an INQUERY inference network. The document portion of the network is computed in advance and the query por-

tion is computed at retrieval time. The document side consists of document nodes $d_1 \ldots d_i$, text nodes $t_1 \ldots t_j$ and concept representation nodes $r_1 \ldots r_k$. The document nodes represent abstract documents. A document may consist of text, figures, images, formatting information, etc. The text nodes represent the textual component of the documents. A given text node corresponds to the observation of the text of a document. For our purposes, we can assume that there is a one-to-one and onto relationship between text nodes and document nodes but, of course, in general that is not necessarily true. Several documents could share a textual component and, conversely, non-textual information can be considered as a another source of information within the same model, but these two cases are not relevant to the present discussion.

The concept representation nodes $r_1 \ldots r_k$ are features that can be possessed by the document texts. A link to an r node means that a document is "about" that particular concept. There is some uncertainty to be resolved due, for example, to differences in word sense e.g., the word "train" may mean that a document is about trains as a mode of transportation or it may mean that document is about training employees, as well as other factors. This distinction between words and concepts is the key distinction to be made here. The uncertainty of indexing in information retrieval is based on not having a direct representation of concepts. Instead, the probabilities of concepts are estimated from word occurrence statistics.

The query side of the network consists of the query concepts, $c_1 \ldots c_m$, some number of queries, q_1 and q_2 in this diagram, and I the information need. The query concepts are the primitive concepts used to represent the query. The concepts will be represented by the words of the query, but again there is a fair amount of uncertainty due to differences in word sense. The query nodes represent individual queries. In this model, the information need of the user can be represented by more than one query, using more than one type of information. The information need itself is known only to the user and needs to be inferred from the queries. There can be an additional layer

between the query nodes and the concept nodes representing intermediate operators such as phrases or Boolean operators. The task then is to calculate the belief in the information need of the user given each document. The documents will then be ranked by their belief scores [61].

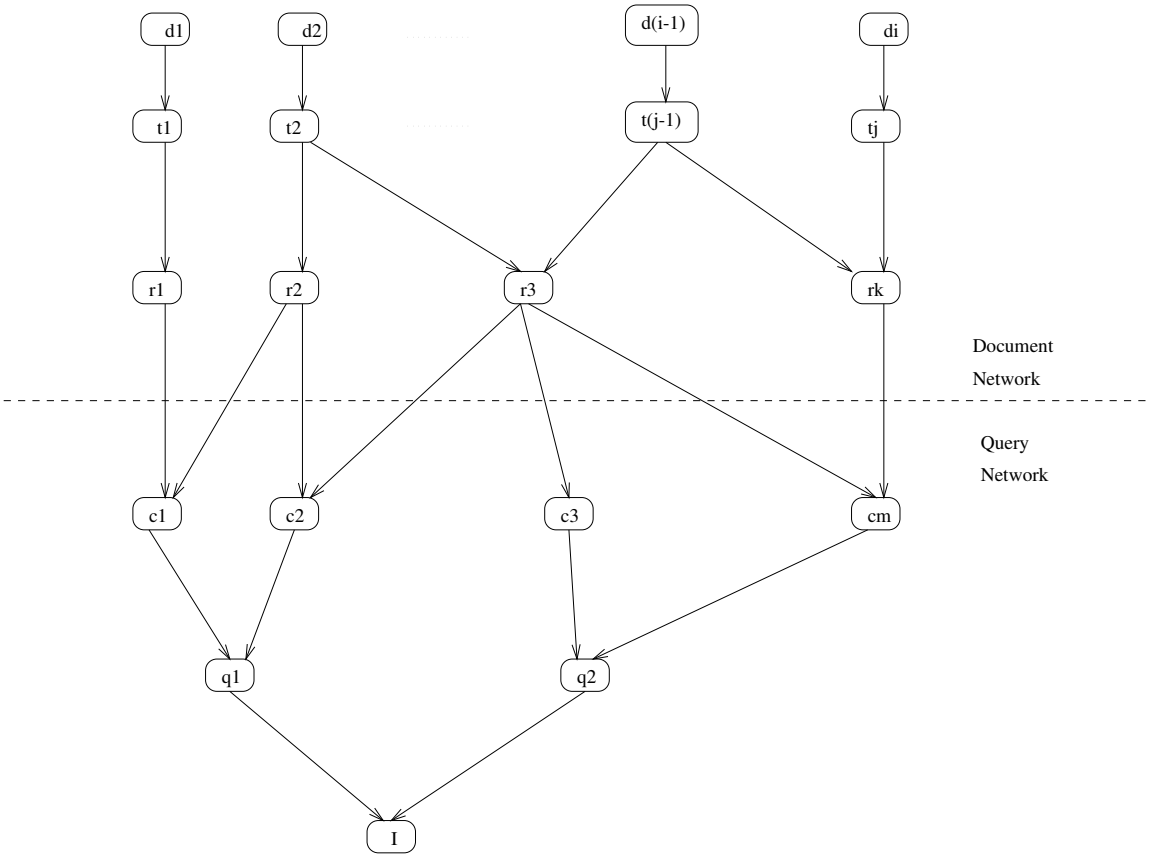

Figure 2.8. Example inference network.

2.2.2.3 A Utility Theoretic and Information Theoretic Approach

Wong and Yao [65] proposed a model in which they represented documents according to a probability distribution. They then developed two separate approaches to retrieval, one based on utility theory and the other based on information theory. Regarding the probability distribution, Wong and Yao use a maximum likelihood estimator for term probabilities. Wong and Yao's utility and information theoretic retrieval models are somewhat analogous to other approaches to retrieval in that

they have an indexing model apart from their retrieval model. Terms are associated with documents according to the maximum likelihood probability estimate and the discriminant is a utility theoretic or information theoretic function of this estimate.

The main problem with the Wong and Yao approach is that the maximum likelihood estimator is not adequate for this task. This point is discussed further in section 2.2.2.4 and a solution to this problem is given in Chapter 4.

2.2.2.4 The Multinomial Model

The most similar approach to the one taken in this thesis is that of Kalt [32]. In the Kalt model, documents are assumed to be generated by a stochastic process: a multinomial model. The task investigated was text classification. Each document was treated as a sample from a language model representing the class of that document. In this model, document length and within-document term frequency are both integral parts of the model rather than being used heuristically as they are in many other models. The discriminant function is taken to be the maximum likelihood estimator of the query terms given the document's language model. Note that the 'query' in this case was inferred from the training set in the context of the classification task.

As with Wong and Yao's approach, the use of the maximum likelihood estimator was the main problem with the Kalt approach. It should be noted that this estimator makes sense given Kalt's view of the problem. This estimator is, in a certain sense, a perfect predictor of each document. However, when one considers the generation of queries, rather than documents, it becomes apparent that better estimation is needed. This is a key point and will be developed further in chapter 4. The major difference between the approach that will be developed in Chapter 4 from the Kalt approach, is that a more robust estimator will be employed in lieu of the maximum likelihood estimator. This is a key difference. Kalt reported results for the TREC-3 routing task that are considerably lower than those that will be described in Section 5.2.2.

An important reason for the difference is that Kalt's probability estimation is not as robust as the method described in Chapter 4.

While the new approach is conceptually somewhat different, it is clearly related to the Kalt approach and shares the desirable property that the collection statistics are integral parts of the model. An additional difference is that Kalt's assumption that documents were drawn from k language models representing the k classes of interest will not be made here. Instead, a weaker assumption will be made, that estimates of each document's language model can be obtained individually without making inferences about the class membership of documents. These models can then be used to compute the query generation probability.

To sum up, there is the vector space model which is simple and intuitive but that does not explain the process of retrieval or how it can be improved. There are also the early probabilistic approaches to retrieval which are explanatory as far as they go, but that do not address the problem of within-document term weighting. Finally, there are the more recent approaches that do explain within-document term weighting but that, in general, do so in a very complex manner. Moreover, these models require the inferences related to the semantics of the text, the probabilities of which must be estimated using heuristics.

A model that does not require heuristics is the Kalt model for document classification. However, the Kalt model did not perform well compared to other approaches to the classification task. The main problem with the Kalt approach lies in the estimation. It will be shown in chapters 4 through 6 that a related model that shares the desirable characteristics of Kalt's model performs very well in comparison to other approaches to retrieval on several tasks largely due to improved estimation.

Also, the new model introduced in this thesis shares the desirable property of simplicity and intuitiveness with the vector space model but it also provides an explanatory model of retrieval, including within document term weighting as provided

by the more complex modern probabilistic models. The simplicity is retained by avoiding the question of semantics altogether and replacing it with the notion of language modeling. It will be shown that the language models can be estimated well enough from the available data to provide effective retrieval.

2.2.3 Extended Boolean Queries

Many modern retrieval systems support Boolean queries. One potential problem with Boolean queries is that users do not always have good intuitions as to how to specify them in order to achieve the desired result. To combat this problem, researchers have proposed generalizations of Boolean operators that give extra-weight to documents that satisfy the Boolean queries exactly but also give reduced weight from other documents. This allows users to gain the benefits of Boolean queries without being hurt by overly specific queries.

2.2.4 The P-Norm Model

An implementation of generalized Boolean operators proposed by Salton *et al.* [53] is the P-Norm model. Recall that in Salton's vector space model of retrieval, documents and queries are treated as vectors in a T dimensional space where T is the number of terms in the collection.

For a two term query with query term weights q_1 and q_2 and document weights d_1 and d_2, the L_2-Norm implementation of OR is computed as follows:

$$\sqrt{\frac{q_1^2 d_1^2 + q_2^2 d_2^2}{q_1^2 + q_2^2}}$$

and similarly, the L_2-Norm implementation of AND is computed as:

$$1.0 - \sqrt{\frac{q_1^2 (1.0 - d_1)^2 + q_2^2 (1.0 - d_2)^2}{q_1^2 + q_2^2}}$$

This assumes that the distances of document values in the vector space is Euclidean. In order to generalize this family of operators, the L_p-Norm for OR is computed as follows:

$$\left[\frac{q_1^p d_1^p + q_2^p d_2^p}{q_1^p + q_2^p}\right]^{1.0/p}$$

and similarly, the L_p-Norm implementation of AND is computed as:

$$1.0 - \left[\frac{q_1^p (1.0 - d_1)^p + q_2^p (1.0 - d_2)^p}{q_1^p + q_2^p}\right]^{1.0/p}$$

This formulation allows generalized versions of AND and OR operators to be incorporated into queries. However, the reason why one should expect the vector norms to improve retrieval effectiveness is unknown. Since there is an additional parameter, one has an additional degree of freedom in order to better fit the data, but the reason the fit is better is not specified by the vector space model. In addition, no principled method is known for choosing a value for p. Instead, values of p are generally set empirically.

2.2.5 The PIC Operators

The P-Norm model does not have a probabilistic interpretation. This makes these operators undesirable for use in the INQUERY inference network model, since they violate the semantics of the formalism. In order to implement a probabilistically motivated family of generalized Boolean operators, Greiff *et al.* [24] developed the PIC operators.

In the INQUERY model, inferences are made as to the presence of concepts in documents. Each concept node represents the degree of belief in that concept for the given document. In order to combine concept nodes, the PIC matrices make the simplifying assumption that only the number of true concept nodes will effect the

final outcome. This is known as the "parent indifference criterion", and is where the PIC operators get their name. By making this assumption, the operators can be computed efficiently by means of a dynamic programming algorithm developed by Greiff *et al.*

The characteristics of each operator is determined by a single parameter that controls the rate of increase as a function of the number of true parents. A range of values for the parameters was determined empirically.

2.2.6 Passage Retrieval

Passage retrieval is a generalization of the text retrieval task where relevant passages rather than entire documents are retrieved. This task is somewhat related to the topic segmentation task and so it will be discussed from that point of view. Passage level evidence using fixed length vs. variable length passages in the context of the document retrieval task will be discussed further in section 2.2.6. Additional discussion of passage retrieval in the context of the language modeling approach to information retrieval will be presented in Section 6.2.4.

Salton *et al.* used features such as paragraph or section boundaries [52] for passage retrieval. The test data for this study was a collection of text from an online encyclopedia. In this well-edited collection, paragraph and section boundaries provide useful information. For the task of topic segmentation, it cannot be assumed that paragraph and section information will be reliable in heterogeneous collections. Moreover, this information is not available at all in broadcast speech data. However, it will be shown that the task can be performed well even without such information.

As mentioned earlier, much of the work on passage retrieval has used passages of fixed length. An exception to this is Mittendorf and Shäuble [39] in which Hidden Markov Models (HMMs) were used to retrieve passages of variable length. This is an interesting approach and is somewhat related to the current work on topic seg-

mentation in that the text is broken up using a sequential (Markov) decision process. The difference is that in their approach, a specific information need is modeled by a stochastic process which generates text fragments relevant to a particular query. This process is called the passage model. A second process generates typical text fragments without regard to any query. This process is called the background model.

In terms of the notation introduced in Section 2.1.5, the A matrix represents the transition probabilities between the background and passage models and the B matrix contains probabilities of words associated with the two models. A text block t is scored as a function of the probability that the passage model produced t and the background model produced the surrounding context of t. This work is related in two ways. First, in the context of segmentation, the approach taken in this thesis is to use HMMs with several background models since, for the task of interest, the information need was not expressed a priori.

Regarding the language modeling approach to retrieval, discussed in Chapter 4, one can view each document as a language model. One could view a collection of documents as a collection of models similar to the collection of states in an HMM. The major point of departure is that sequential effects are not modeled for the retrieval task.

Callan [12] used passage level evidence to improve results for ad hoc retrieval and compares the effectiveness of discourse and window passages. Discourse passages could be any identifiable units of discourse but are restricted to single paragraphs in Callan's paper. Experiments were also done with bounded paragraphs where paragraphs are used as passages but with minimum and maximum length requirements. Results of experiments on four test collections show that using paragraph or bounded-paragraph passages yields results worse than the baseline system but passage level evidence in combination with the baseline system improves results slightly. On the other hand, fixed length passages alone yielded better results than the baseline sys-

tem and results were better still when used in combination. However, for different test collections, different passage sizes were optimal.

Allan [1] used passages in the context of information routing. Rather than performing relevance feedback on entire documents, only the best passage was used. Results show that this method is more effective than feeding back entire documents. In this study passages were of fixed length and experiments were done with passage sizes ranging from 50-1000 words.

Hearst and Plaunt [30] used variable length passages, which they call tiles to augment ad hoc retrieval. They report a significant increase in performance but also report a similar increase using paragraph passages. They did not perform experiments using fixed length passages. A natural question in light of Callan's results [12] is whether fixed length passages would have been better still. Their larger improvement (relative to [12]) may have been due to a lower baseline.

Knaus *et al.* [33] use passage level evidence to improve ad hoc retrieval. Passages in this study are variable length. They report a ten percent improvement in average precision over the basic document level method.

The question of whether passage level evidence is useful for document retrieval remains an open question. With the introduction of Robertson's *tf*, and the corresponding higher baseline performance, the improvements provided by passage level evidence have largely disappeared. This is not an intuitive result and cannot adequately be explained. In Section 7.2.6, this question, and suggestions for how to address it, will be discussed further.

2.2.7 Relevance Feedback

As noted in Section 1.1, the technique of relevance feedback can be used to improve retrieval effectiveness. In an interactive setting, a user can run a query and mark one or more documents in the resulting ranked list as relevant to the query. At this point,

the system will choose terms and possibly other features such as phrases to add to the query based on the occurrence statistics in the marked documents. In addition, the query features will sometimes be re-weighted.

2.2.7.1 The Harper and van Rijsbergen Model

In 1978, Harper and van Rijsbergen developed a method for using relevance information, obtained by relevance feedback, to obtain better estimates for the probability of relevance of a document given the query. This work attempted to correct for the assumption of independence which the authors did not think was realistic [27].

Given complete relevance information, an approximation of the dependence of query terms was defined by the authors by means of a maximal spanning tree. Each node of the tree represented a single query term and the edges between nodes were weighted by a measure of term dependence. Rather than computing the fully connected graph, the authors computed a tree that spanned all of the nodes and that maximized the expected mutual information computed as follows:

$$\sum_{i,j} P(x_i, x_j) \log \left(\frac{P(x_i, x_j)}{P(x_i)P(x_j)} \right)$$

where i and j range over the query terms, $P(x_i, x_j)$ is the probability of term x_i and term x_j occurring in a relevant document, $P(x_i)$ is the probability of term x_i occurring in a relevant document and, likewise, $P(x_j)$ is the probability of term x_j occurring in a relevant document. Note that these probabilities refer to the probability of occurrence of the term or term pair one or more times in a document. The within document frequency is not accounted for by this measure.

Harper and van Rijsbergen did two sets of experiments using this approximation. The first was to determine the upper bound performance of the term dependence model vs. a term independence model using the complete relevance judgments. These experiments showed that under these conditions, the dependency graph did, in fact,

yield useful information and resulted in more effective retrieval than the model that did not utilize this information.

The second set of experiments used the dependency graph in the presence of a limited number of relevance judgments; both ten and twenty judged documents were tested. In order to compensate for the relatively sparse data, the authors used a Bayesian estimator with Jeffrey's prior as recommended in [62], to estimate the probabilities. The authors found that with a limited number of judgments, it was still the case that the dependency graph model yielded better retrieval. It should be noted that the collections used for these experiments were quite small, as larger collections were not available at the time, and the authors cautioned against drawing firm conclusions from these results [27].

2.2.7.2 The Rocchio Method

A commonly used method of relevance feedback is due to Rocchio [49]. The Rocchio method provides a mechanism for the selection and weighting of expansion terms and is defined as follows:

$$w_r(t) = \alpha w(t) + \beta \frac{1.0}{|R|} \sum_{R} w(t) - \gamma \frac{1.0}{|\overline{R}|} \sum_{\overline{R}} w(t) \tag{2.1}$$

where α is the weight assigned for occurring in the initial query, β is the weight assigned for occuring in relevant documents, γ is the weight assigned for occurring in non-relevant documents, $w(t)$ is a weighting function, generally based on term frequency and/or document frequency, R is the set of documents judged relevant and $\overline{R}$ is the set of documents judged non-relevant. This formula can be used to rank the terms in the judged documents. The top N can then be added to the query and weighted according to the Rocchio formula.

This is a reasonable solution to the problem of relevance feedback that works well in practice. However, there is no principled way to determine the optimal values of

α, β and γ and so these parameters are generally set empirically. Also, the weighting function, $w(t)$ is based on heuristic use of the collection statistics. The method described in Section 5.2 will be derived from the language modeling approach and will not require heuristics and empirically set parameters.

2.2.7.3 Relevance Feedback in the INQUERY Model

The formulation of the INQUERY inference network model in 1991 by Turtle [61], included discussion of relevance feedback. The theoretical work of actually adding relevance feedback to the inference network model and the implementation of the theoretical ideas was done in 1996, by Haines [17]. A new type of node was added to the inference network to reflect the relevance judgments provided by the user. These annotation nodes were added to the query side of the inference network and the evidence from the annotations was propagated through the network using message passing.

Figure 2.9 shows a closeup view of a portion of the query network from Figure 2.8. Once again, $c1$ and $c2$ represent query concept nodes, $q1$ represents a query and I represents the information need of the user. The relevance judgments will be represented by means of a set of annotation nodes.

In order to incorporate annotations, each query concept node is annotated using three additional nodes as shown in figure 2.10. The nodes $k1$ and $k2$ represent the proposition that nodes $c1$ and $c2$ imply that the information need I has been satisfied. The nodes $j1$ and $j2$ represent the observed relevance judgments. The *and* nodes are used to require that the query concept occur in the document in question in order for an annotation to have an effect on the score.

Haines was able to show that an inference network annotated in this way can be computed efficiently [17], thereby showing that relevance feedback can be incorporated into the inference network model in practice as well as in theory.

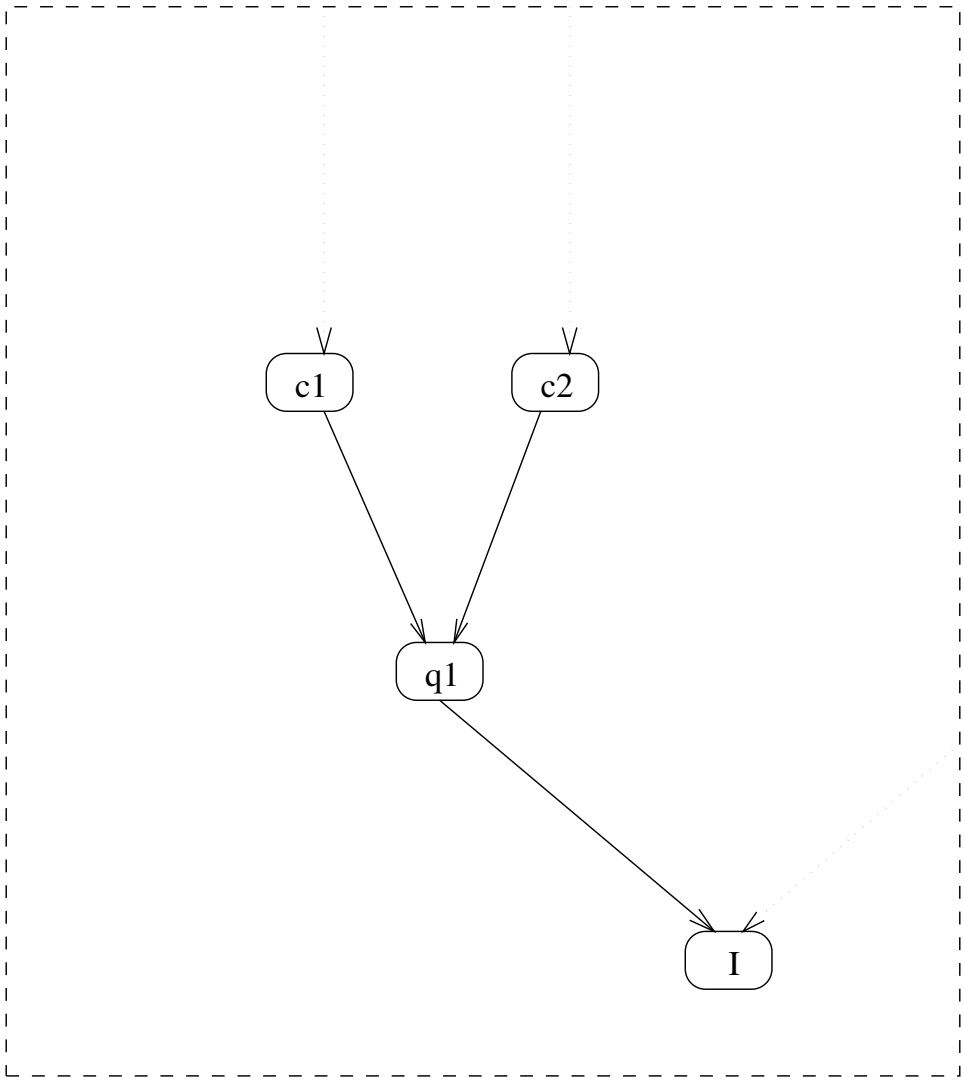

Figure 2.9. Close-up view of query network.

The only drawback of this technique is that it requires inferences of considerable complexity. In order to make use of relevance judgments, two additional layers of inference and several new propositions are required. So, while this method has been shown to work, it is more complex than is desirable. The required node structure is not obvious nor are the implications of this node structure for the improvement of relevance feedback.

In contrast, the method of relevance feedback that will be described in section 5.2 follows more directly from the language modeling approach described in Chapter 4.

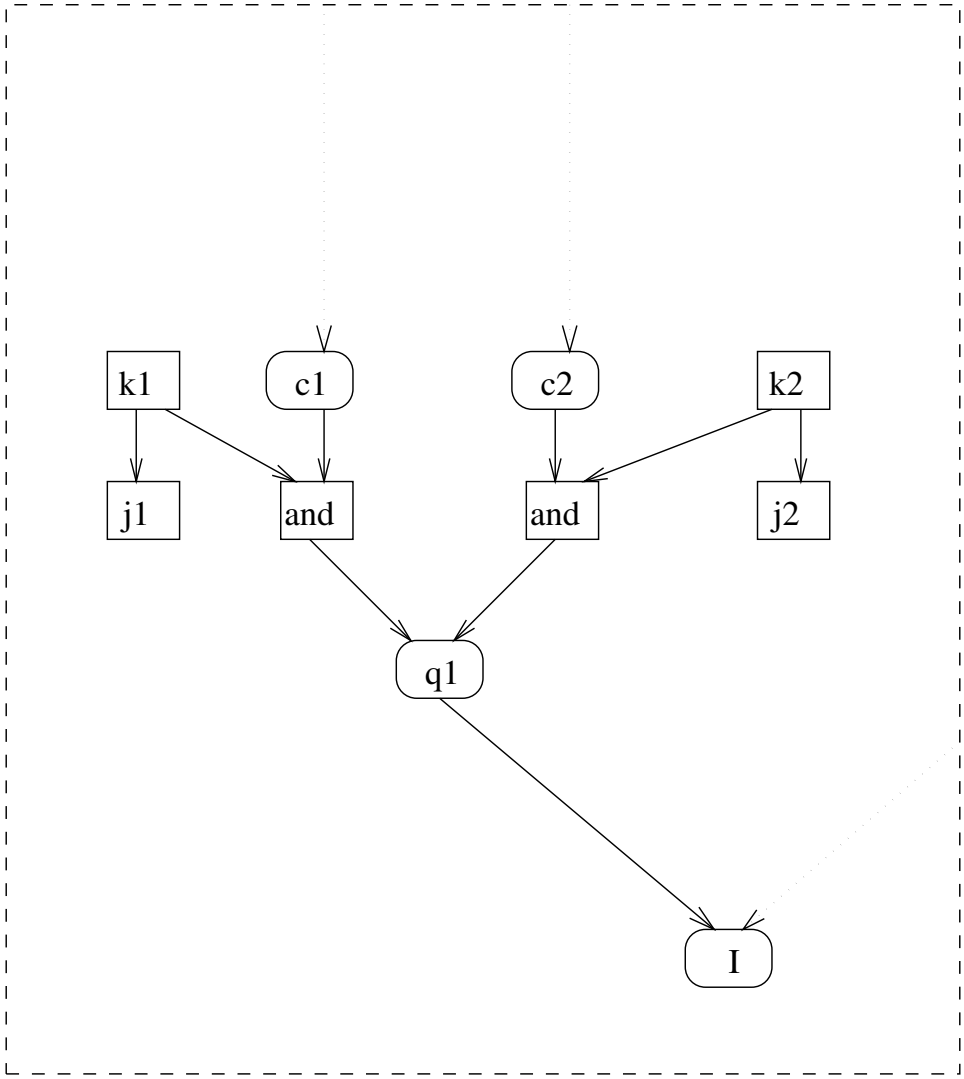

Figure 2.10. Annotated query network.

This technique is very straightforward and will be shown to work as well as existing methods of relevance feedback.

2.2.8 Query Expansion Without Relevance Information

One source of problems in IR is the mismatch of query and document features. For example, one might wish to find documents about connectionist models, but the query "connectionist models" would miss documents that only use the phrase "neural networks." A similar problem arises in the text segmentation task. Within-

topic sentences may not have enough words in common to allow the determination to be made that they are, in fact, discussing the same topic. Researchers have tried several approaches to fix the problem of document/query mismatch. Several of these techniques are discussed here. The study described in Chapter 3 shows that one of these techniques, Local Context Analysis, is useful for the segmentation task.

2.2.8.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) [50] is a technique intended to improve performance of the vector space retrieval model. In the vector space model [51], documents and queries are represented as vectors in an N-dimensional vector space where N is the number of indexing features, usually the number of unique words in the collection after stemming. LSI attempts to reduce the dimensionality of this space by singular value decomposition using co-occurrence information, i.e., the feature space is aggregated into higher order concepts. The result of the decomposition is a lower dimensional space which approximates the original feature space. Query terms can then be expressed as a linear combination of the higher order features. The technique is extremely computationally expensive and, while it has the effect of improving recall, it does so at the expense of precision as reported in [20].

The effect of LSI is to enhance recall. For example, the query "plane crashes" might also return documents describing helicopter crashes if the co-occurrence statistics of "helicopter" and "plane" causes LSI to treat them as a higher order concept. The major problem with LSI is that it can have negative effects on precision since information is lost. In the previous example, one may not be interested in helicopter crashes but, with LSI, it is no longer possible to make that distinction. Dumais [20] shows examples where this effect causes problems. Due to the time complexity, brittleness and adverse effects on precision, as well as the potential dynamic nature of the data, we do not consider LSI a viable option for the segmentation task.

2.2.8.2 An Association Thesaurus

A different approach, taken by Jing and Croft [31], is the automatically constructed association thesaurus. As in LSI, co-occurrence information of concepts is examined. In the case of the association thesaurus, called PhraseFinder, concepts consisting of nouns and noun phrases are extracted from the collection. Each concept is indexed by the words that co-occur with it in the collection. This is done by constructing pseudo-documents where the concept is the title of the pseudo-document and the co-occuring words are the text. At query time, the original query will be run against the PhraseFinder pseudo-document database and the concepts, the titles of the top retrieved pseudo-documents, returned by that query are added to the original query and weighted according to their rank position. This approach is more robust than LSI in terms of retrieval effectiveness and more flexible since it operates at query time. One drawback of this technique is that it is quite resource intensive to construct the association thesaurus.

2.2.8.3 Local Feedback

A technique studied by Attar and Fraenkel in the late '70's [5] is known as local feedback. In this technique, one assumes that the top N ranked documents are likely to be relevant, and expands the query much like relevance feedback. One problem with this technique is that if initial retrieval results are poor, this technique will make them worse. Despite some early success, this technique was largely abandoned for several years.

Recently, however, local feedback was revisited by Buckley *et al.* [10]. In the TREC 4 evaluation, local feedback was used successfully to improve results in the ad hoc retrieval task. In one respect, this is not surprising. Since IR systems perform much better now than formerly, one would expect that the top N documents are more likely to be relevant. Nevertheless, local feedback is still a somewhat risky technique

and is quite sensitive to the value chosen for N, where N is number of top documents selected for feedback [67]. One advantage to local feedback is that is less resource intensive than either LSI or PhraseFinder.

2.2.8.4 Local Context Analysis (LCA)

Local Context Analysis (LCA) is a technique recently developed by Xu and Croft [67]. This technique, like PhraseFinder and local feedback, is used at query time to expand the initial query. Like PhraseFinder, concepts consist of words and phrases. Concepts associated with the initial query are added. Unlike PhraseFinder, only features from the top retrieved documents will be considered, i.e local information as opposed to collection-wide information. LCA is efficient in both time and space complexity compared to LSI and PhraseFinder. It is more robust than local feedback with respect to the number of documents/passages to consider. Most importantly, LCA outperforms all of these techniques. These attributes make LCA the technique of choice to find features for the text segmentation task.

2.3 Text Segmentation

2.3.1 Text Segments and Text Themes

Salton and Singhal [54] and Salton *et al.* [55], discuss the decomposition of text into segments and themes where a segment is a contiguous block of text discussing a single subtopic and a theme is a chain of such segments possibly interleaved with other themes. The segmenting process is done at the paragraph level by computing cosine similarity of adjacent paragraphs.

2.3.2 Text Tiling

Hearst [29] and Hearst and Plaunt [30] discuss a method of segmenting expository texts into multi-paragraph subtopics which they call 'tiles' using cosine similarity in conjunction with smoothing. First, they break up texts into blocks of size N (where

N ranges from three to five sentences in their experiments). Next they compute a similarity curve of adjacent blocks using cosine similarity and smooth this curve to get rid of local extrema. Finally, they use the resulting smoothed graph to identify potential topic boundaries. Each local minimum corresponds to a topic boundary. They performed experiments using these topic boundaries to improve ad hoc retrieval effectiveness and also compared the segmentation to that of human judges. While the agreement to the human judgments was reasonable, the improvements in retrieval performance were not better than those achieved using fixed length passages.

2.3.3 Multiple Language Models

Yamron *et al.* [3] developed a segmenter based on Hidden Markov Models (HMMs). Their approach was to construct one hundred language models, each model roughly corresponding to a topic. They produce these language models using a training corpus containing story boundaries. First they cluster the stories using a k-means clustering algorithm and then they produce a unigram language model for each of the resulting clusters.

The clustering algorithm works as follows:

1. Set a clustering threshold t.
2. For each story, place it in the closest cluster (as defined below).
3. If the distance is greater than t, create a new cluster.
4. Repeat the above process several times, taking each story out of its current cluster and re-clustering. Clusters are added as needed and removed if they become empty.

The threshold t was not specified in [3] but seems to have been determined empirically. The distance metric was defined as follows:

$$d = \sum_{w_i} (s_i / \sum s_i) \log \frac{s_i / \sum s_i}{(c_i + s_i)/(\sum c_i + \sum s_i)} + (c_i / \sum c_i) \log \frac{c_i / \sum c_i}{(c_i + s_i)/(\sum c_i + \sum s_i)}$$

where s_i and c_i are the story and cluster counts for word i. They refer to this distance as a variation of the symmetric Kullback-Leibler (KL) metric. Apparently this measure is intended to insure that words will have a distribution in the clusters similar to their distribution in the stories of the training data, though this point is not discussed in [3].

The resulting language models are then smoothed using the smoothing method described in Section 2.1.5 and each model is used as a state in an HMM. The A matrix in this HMM is the probability of seeing a particular language model given the previous model. The B matrix is the term distribution for each model, i.e., the probability distribution over words in the corpus for each of the one hundred models.

The task of segmenting is then carried out by using the Viterbi algorithm to determine the maximum likelihood sequence of models and then inserting a break each time there is a transition to a new model [3].

2.3.4 Exponential Models

Beeferman *et al.* [8] describe an approach to segmentation using exponential models. The model utilizes two types of features which they call 'relevance features' and 'lexical features.' A trigram model as described in section 2.1.3 is used as the short range model. Longer range 'triggering' effects are incorporated into the model by means of a recently seen word cache and then the probabilities can be calculated using an exponential model as described in Section 2.1.4. To be more precise, the relevance features consist of ratios of probabilities of long range language models and short range language models as follows:

$$L = \log \left(\frac{\left(\frac{e^{\lambda \cdot f(x)} p_{tri}(x)}{\sum_X e^{\lambda \cdot f(x)} p_{tri}(x)} \right)}{p_{tri}(x)} \right)$$

where the features of exponential model f_i consist of words that predict the word x within a predefined window of 500 words and $p_{tri}(x)$, the trigram probability, is the probability of x given the two previous words. The intuition behind this approach is that when a long range language model is not able to predict the next word better than a short range language model, that indicates a shift in topic. In Section 5.2, a similar ratio based approach will be used for a different purpose, prediction of useful terms for relevance feedback.

The second class of features, the lexical features, consisted of words whose presence predicts a break or the absence of one. An example of such a feature is the personal pronoun 'he.' The presence of the word 'he' indicates that the current sentence is probably not the start of a new topic since it refers to a person introduced previously in the story. The notable characteristic of their approach is that these features were induced from the training data automatically using a sophisticated feature induction algorithm. However, the space of possible features was determined by Beeferman *et al.* in advance. The features are induced one at a time and their weights are calculated using the improved iterative scaling algorithm [18].

This algorithm is defined in a very general sense in [18]. The specifics of the algorithm for the segmentation task are given here.

Once again, the exponential model is defined as follows:

$$\hat{p}(x) = \left(\frac{e^{\lambda \cdot f(x)} p_d(x)}{\sum_X e^{\lambda \cdot f(x)} p_d(x)} \right)$$

where λ is a vector of parameters, $f(x)$ is a vector of features that predict x and $p_d(x)$ is a default probability estimate of x, the probability of seeing a break at the

current position. The feature vector $f(x)$ can consist of a combination of the relevance features and lexical features described above. The features are chosen one at a time according to the maximal gain criterion defined as follows:

Let B_p and B_q be binary valued random variables defined as:

$$\begin{aligned} B_p(0) &= 1.0 - \tilde{p}[g] \\ B_p(1) &= \tilde{p}[g] \\ B_q(0) &= 1.0 - q[g] \\ B_q(1) &= q[g] \end{aligned}$$

where $\tilde{p}[g]$ is the expected value of candidate feature g in the training data and $q[g]$ is the expected value of g according to the current model q. Having defined these variables, the maximal gain is computed as follows:

$$\hat{G} = -\sum_{0,1} B_p \log\left(\frac{B_p}{B_q}\right)$$

This is the Kullback-Leibler divergence of the current model with the observed data. In other words, it is the maximal degree to which the model can be improved by feature g. This is a greedy search in the following sense, the gain is maximal using the current values of the λ vector. To find the feature with the real maximum gain, the λ vector would need to be re-estimated, as described below, for each candidate feature. The greedy algorithm is used to avoid performing the costly re-estimation algorithm for every candidate feature.

Once a feature has been chosen, it is assigned the value, $\hat{\alpha}$ defined as:

$$\hat{\alpha} = \log\left(\frac{\tilde{p}[g](1.0 - q[g])}{q[g](1.0 - \tilde{p}[g])}\right)$$

as an initial estimate for its λ value where $\hat{\alpha}$ is the closed form solution for the value that produces the maximal gain. For the theoretical derivation of $\hat{G}$ and $\hat{\alpha}$ see [18].

After choosing a new feature and assigning the initial value of λ for the new feature, the entire λ vector is re-estimated by solving the following equation for each λ_i:

$$q\left[f_i e^{\lambda_i f_\#}\right] = \tilde{p}\left[f_i\right]$$

where $f_\#$ is the sum over the observed training data of the number of features that are 'on' for a given observation. As before, the square brackets indicate the expectation of the variable under the indicated distribution, where q refers to the distribution determined by the exponential model and $\tilde{p}$ is the empirical distribution ofthe feature in the training data itself. For the segmentation work, this equation was solved using Newton's method [37]. The process of adding the maximal gain feature and then re-estimating is repeated until the maximal gain of all candidate features is below an empirically determined threshold.

CHAPTER 3

LANGUAGE MODELS FOR TEXT SEGMENTATION

This chapter begins with a brief introduction to the segmentation problem in Section 3.1. Next, an overview of the TDT project, the project that motivated the segmentation work, is given in section 3.2, along with a description of the evaluation methodology used later in the experiments.

Next, in Section 3.3, the approaches to segmentation are described along with the features that will be used. In addition, a preliminary study is described showing how local context analysis (LCA) can be used to find some of the features.

Section 3.4 describes how the two feature sets, previously described in Section 3.3, can be combined into a single probabilistic model. The empirical results of segmenting with this combined model are then presented in Section 3.5. The chapter concludes with a discussion of the results in Section 3.6.

3.1 Newsfeeds and Topic Boundaries

There has recently been interest in tracking events in information "feeds." A feed is a continuous stream of text produced, for example, by speech recognition of broadcast news. The text in such an environment contains no mark-up to indicate topic boundaries and may not even delineate sentences. The text also tends to be terse and words are not repeated to the degree they would be in, for example, an academic journal paper. Approaches to text segmentation that depend on redundancy at the word level such as [29] will fail on such data.

The work presented here was done as part of the Topic Detection and Tracking (TDT) pilot study [3].

3.2 The TDT Tasks

The TDT pilot study evaluated the state of the art in three tasks, new event detection, event tracking and segmentation. For our purposes, the TDT segmentation task will be explained and also the event tracking task since it is relevant to the indirect evaluation of the segmenter.

The TDT study corpus consisted of approximately 16000 news stories. Approximately half came from CNN news transcripts and the remaining half were from the Reuters newswire.

3.2.1 Segmentation

The input to the segmentation module is a stream of text without explicit topic or document boundaries. The task is to break this text into blocks representing homogeneous topics. The ability to do so is useful in any case where a stream of text does not contain explicit document or topic markers such as collections of automatically recognized speech or collections of text documents in which the documents contain multiple topics. In such cases, segmentation is an enabling technology for the event detection and tracking tasks as well as any similar tasks, such as document routing or filtering. In the context of the TDT evaluation, the story boundaries were removed from the study corpus and the task was to predict story boundaries using an automatic segmenter. [3]

3.2.2 Event Tracking

One method of evaluation for the segmenter is the evaluation of the effects of segmentation on a real-world task. The task that will be used is the tracking of events. The event tracking task can be described as assigning labels to stories from a

predefined set of topic labels [3]. If one views this task in relationship to the document routing task described earlier, it is very similar but it has the added dimension of time and the task centers on specific events rather than more general topics as in the routing task. Indirect evaluation of the segmenter in the context of this task will be described in section 3.2.4.

3.2.3 Segmentation Direct Evaluation

3.2.3.1 A Probabilistic Error Metric

The segmenter can be evaluated directly by computing the probability that two words drawn at random from the corpus will be correctly classified as to whether they belong to the same segment. However, an algorithm should get less credit for correctly predicting this information regarding the first and last words in the corpus, which are almost certainly in different segments, than for correctly predicting for words near each other. This can be accomplished by a kernel type estimator originally proposed by Beeferman *et al.* [8].

The error probability is defined as follows:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} D(j-i) \times \delta_r(i,j) \overline{\oplus} \delta_h(i,j)$$

where, N is total number of words in the corpus, $\delta_r(i,j)$ equals 1 if words i and j belong to the same segment and 0 otherwise. Similarly, $\delta_h(i,j)$ equals 1 if words i and j have been predicted by the segmenter to belong to the same segment and 0 otherwise. $\overline{\oplus}$ means exclusive NOR, meaning that either both or neither of the δ functions must evaluate to one. Finally, $D(j-i)$ is the distance probability distribution which weights words that occur in close proximity more heavily than those further away. The distribution originally proposed for D by Beeferman *et al.* was a single parameter exponential distribution. However, for the TDT evaluations, the distribution chosen

by Doddington [19] was a triangular distribution with standard deviation equal to one half the average segment length in the study corpus, approximately 250 words.

3.2.3.2 Recall and Precision

In addition, the metrics of recall and precision were employed for a preliminary study. To compute recall and precision, first a least squares alignment of the predicted segmentation with the correct segmentation is performed. Then the distance between the two segmentations is measurable in terms of *insertions*, *deletions* and *moves*. An *insertion* error occurs when the algorithm produces a break that does not line up with a real break when the least squares alignment is done. Similarly, a *deletion* error occurs when a real break exists but the algorithm does not produce a break that lines up with it. Finally, a *move* error occurs when two breaks line up, but are not in the same place. Table 3.1 shows an example. The first column shows the true segmentation and the second column shows the predicted segmentation. In this example, an *insertion* error occurs at sentence 5 where the algorithm has predicted a break which does line up with a real break. A *deletion* error occurs at sentence 27 where the original segmentation has a break which is not found by the algorithm. Finally, two *move* errors occur at sentences 26 and 31 where the algorithm has predicted breaks when the real breaks are at sentences 25 and 29.

Table 3.1. Example of aligned segmentations.

Actual Segmentation	Predicted Segmentation
2	2
	5
10	10
14	14
19	19
25	26
27	
29	31
35	35

3.2.3.3 The Pessimistic Error Function

Given the aligned segmentations one can choose an error function for the *move* errors based on the desired level of tolerance. An *insertion* or *deletion* always counts as one error. Results are reported using two error functions. The first is a 'pessimistic' error function:

$$f(b) = \begin{cases} 1 & \text{if } \exists r(b = r); \\ 0 & \text{otherwise} \end{cases}$$

This function simply counts each break b if and only if it matches some real break r exactly. This measure does not take into account predictions that were close but not exact, e.g. a block of length six would count as a complete miss even if the real block started in the same place and was of length seven.

3.2.3.4 The Partial Match Error Function

The problem with the exact match error function is that it does not distinguish between some segmentations where it clearly should. For example, suppose a data set consists of one hundred sentences with a topic break at sentence fifty. The exact match error function would not distinguish between an algorithm that predicted a single break at sentence forty-nine and one that predicted a single break at sentence two even though the former is clearly better. However, the following error function does make that distinction:

$$f(b) = \left(\frac{1.0}{2d}\right) \times \left(\frac{((u-g)-d)}{u}\right)$$

In this function, d is the difference between the predicted break b and the corresponding real break (where correspondence is determined by the alignment process), u is the difference between the next actual break and the previous one, and g is the number of insertion errors between the next actual break and the previous one. It may help to think of u as the amount of uncertainty and g as the number of guesses.

This partial match error function gives full credit for exact matches and partial credit for near misses. The first term causes the amount of credit to drop off as the distance increases and the second term measures the 'nearness' of the near miss.

3.2.3.4.1 Preliminary Study For the purposes of the preliminary study, scores with both of the above error functions are reported. The errors are counted using both error functions and then the counts are used to calculate recall and precision scores for each function. Recall is measured as the percentage of topic breaks in the original data that were predicted by the algorithm. Precision is measured as the percentage of breaks that were predicted by the algorithm that appeared in the original data.

The two sets of recall and precision scores allow more meaningful comparisons than either set by itself. Two segmentations can be compared by the worst case analysis using the exact match score. The partial match score provides a reality check if the exact match scores are close. Also, for a single segmentation, the partial match scores can be compared to the exact match scores to determine the closeness of missed breaks.

3.2.4 Segmentation Indirect Evaluation

The indirect segmentation evaluation is carried out by comparing the results of the event tracking task using the automatically predicted story boundaries as compared to the same task run with the original story boundaries. This provides a more "real-world" measure of segmentation quality as it shows the effects of segmentation on a task of interest.

3.2.4.1 Event Tracking Evaluation

The TDT pilot study evaluated systems on the tasks of new event detection and event tracking. Briefly, the task of new event detection was the identification of the first instance of a story discussing a particular event and the tracking task was the

identification of subsequent stories describing the events of interest. 25 events were tracked. Each story was labeled with a YES or NO label for each of these 25 events. For the tracking task, when the system fails to label a story with the appropriate event label, this results in a miss error. On the other hand, if the system labels a story with an incorrect event label, that results in a false alarm error. These two types of errors can be plotted against each other to evaluate system performance across a range of threshold values in what is known as a Detection Error Tradeoff curve, or DET curve [19]. Figure 3.1 shows an example DET curve. The DET curve plots the estimated tradeoff between miss rate and false alarm rate. The axes are scaled according to a normal distribution of the two metrics. Note that perfect performance for the tracking task would be a single point at the origin. For the indirect segmentation evaluation, one can define perfect segmentation performance as an identical DET curve for the predicted breaks as was obtained by the true breaks [19].

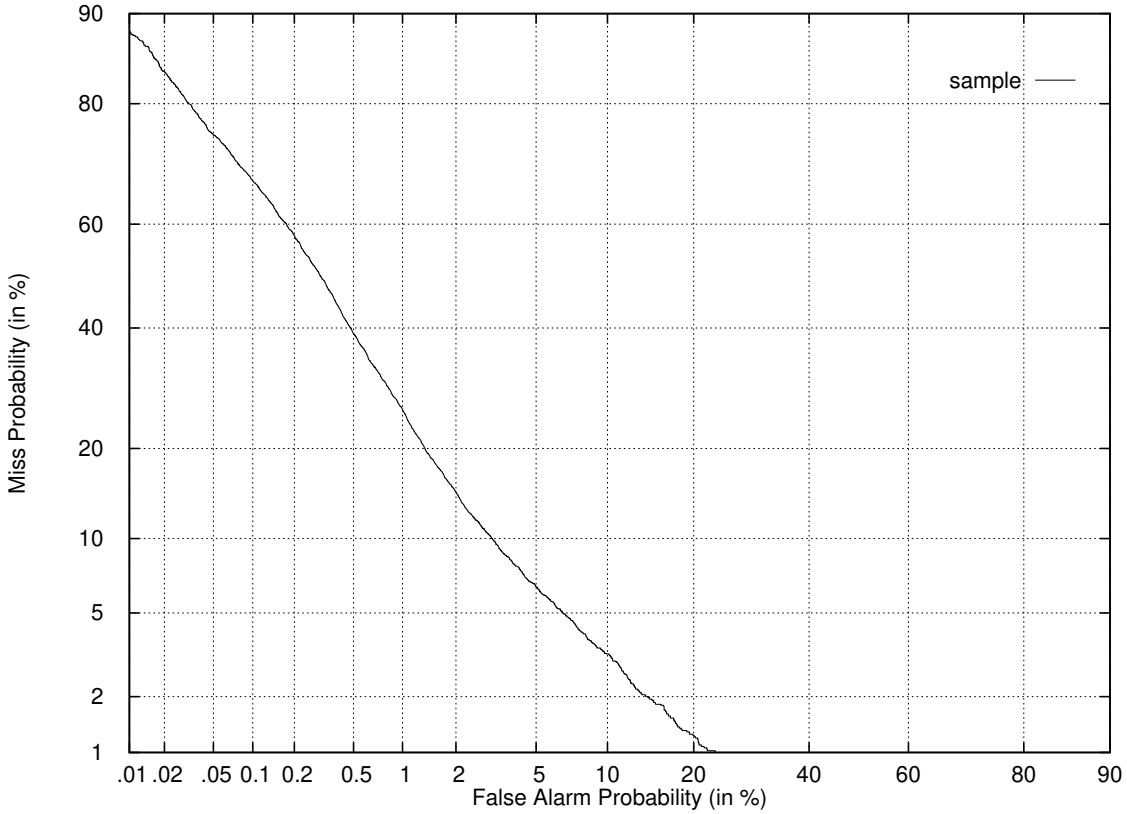

Figure 3.1. Example DET curve.

3.3 Two Complementary Approaches to Segmentation

3.3.1 Content Based LCA Segmentation

For the TDT pilot study, initially two largely complementary segmentation methods were developed. The first method makes use of the technique of local context analysis (LCA) [67]. LCA was developed as a method for automatic expansion of ad hoc queries for information retrieval. It is somewhat like the method of local feedback [15] but has been shown to be more effective and more robust. For the segmentation task, LCA can be thought of as an association thesaurus which will return words and phrases which are semantically related to the query text and are determined based on collection-wide co-occurrence as well as similarity to the original sentence. Each sentence is run as a query against the LCA database and the top 100 concepts are returned. The original sentence is then replaced with the LCA concepts and the effect is that sentences which originally had few or perhaps no words in common will typically have many LCA concepts in common.

3.3.2 LCA Expansion – A Preliminary Study

Experiments were done to test the utility of LCA expansion for content based segmentation [43]. The test data consisted of three sets drawn at random from the "What's News" articles from Wall Street Journal 1989. Test set 1 consists of 228 sentences and 86 segments, set 2 consists of 251 sentences and 96 segments and set 3 consists of 269 sentences and 119 segments. The results were measured using the recall and precision metrics defined in Section 3.2.3.2

3.3.2.1 Results

The results on the three data sets can be seen in Tables 3.2. Note that use of LCA improves the results in all three cases.

To see why this is the case, it will be useful to consider the example in Figure 3.2. Three segments of two sentences each are shown delimited by groups of dashes.

Table 3.2. Comparison of content based segmentation with and without LCA expansion on set 1.

	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
Set 1				
Original Words	70.0%	62.9%	77.9 %	70.1%
LCA Concepts	88.8%	82.6%	91.4%	85.1%
Set 2				
Original Words	58.8%	61.3%	66.8%	69.7%
LCA Concepts	78.4%	79.2%	79.9%	80.7%
Set 3				
Original Words	58.3%	70.7%	64.6%	78.3%
LCA Concepts	75.0%	85.7%	76.6%	87.6%

The important point to notice about this example is the number of words in common in the within-segment sentences. Figure 3.3 shows the counts of non-stopwords in common between pairs of sentences. Notice that the two sentences in the first segment have only one word in common. Likewise for the third segment. The two sentences in the second segment have no common words at all.

A similar table is shown in Figure 3.4 for the same six sentences after LCA expansion. Now the three segments have 10, 11 and 65 concepts in common, making the content based segmentation much easier.

3.3.2.2 The Offset Heuristic

The original LCA method was derived from that described in [43]. The text is indexed at the sentence level using offsets to encode the positions of the LCA features. For example, suppose the feature "O. J. Simpson" occurs in sentence 1, 3, and 10. The index will encode these positions as 1, 2 and 7, the offset from the previous occurrence of the concept. The main idea of the LCA segmenter is to use these offsets to measure shifts in vocabulary over time. The original method, which was tested on the Wall Street Journal in the preliminary study, used a simple function of

```
Police in Lebanon said that two Red Cross workers abducted last week
are being held by Palestinian guerrillas led by terrorist Abu Nidal.
Meanwhile, thousands of students returned to classes as Lebanon's
state schools, most private schools and the American University of
Beirut reopened after being closed for six months.
---
The White House said that a cyst removed Friday from Bush's right
middle finger wasn't cancerous.
A presidential spokesman said a routine pathological examination was
performed on the half-inch growth following the 25-minute surgical
procedure at Walter Reed Army Medical Hospital.
---
Singapore's Lee Kuan Yew said he would step down as prime minister by
the end of next year, ending three decades of nearly one-man rule in
the island nation.
The 66-year-old Lee, in an interview with the British Broadcasting
Corp., said he would hand over power to his deputy, Goh Chok Tong.
```

Figure 3.2. Example of 3 typical topic segments.

the offsets as a heuristic measure of the "surprise" of seeing a particular concept in a particular sentence. A large number of repeated concepts, i.e., small offset values, indicate topical cohesion while new concepts, i.e., large offsets, indicate a new topic.

3.3.2.3 The TDT Corpus

In a collection of newspaper text from a single source, in this case the Wall Street Journal, the offset heuristic in conjunction with LCA expansion worked quite well, as

Sentence	1	2	3	4	5	6
1	—	1	0	0	0	0
2	1	—	0	0	0	0
3	0	0	—	0	0	0
4	0	0	0	—	0	0
5	0	0	0	0	—	1
6	0	0	0	0	1	—

Figure 3.3. Non-stopwords in common from previous example.

Sentence	1	2	3	4	5	6
1	—	10	0	1	0	0
2	10	—	0	0	1	3
3	0	0	—	11	1	1
4	1	0	11	—	0	0
5	0	1	1	0	—	65
6	0	3	1	0	65	—

Figure 3.4. Number of common LCA features for six example sentences.

shown in the preliminary study. However, the TDT corpus has stories from several sources, and so it often happens that several stories on the same topic will occur in close proximity. Moreover, since much of the TDT corpus consists of transcribed speech, there is far more off-topic language than in the Wall Street Journal. For example, throughout the corpus, one finds social interaction between speakers that does not relate to the current topic such as the following exchange between two news reporters:

```
That is where there are-

-Very much so Bob-

-Go ahead Bill.

-Yeah Bob, that's very well said.
```

These two difficulties, repeated topics and off-topic language, were circumvented by means of an exponential length model. Rather than looking at the total size of the offset, a model of the average segment size was used. The model was used to determine the probability that an occurrence of a concept was in the same segment as the previous occurrence. This method is more robust with respect to multiple stories on same topic and also to "social noise" than the original method and performance is improved.

3.3.2.4 Pros and Cons of the LCA method

The LCA method can be thought of as a content-based method. It works by looking at changes in content-bearing words. It is somewhat similar to the topic models used by Yamron *et al.* [3] and to the relevance features used by Beeferman *et al.* [8]. The strong point of the LCA method in comparison to these other approaches is that, other than the length model estimation, it is completely unsupervised. One weakness of this method is that the current implementation is somewhat slow since it requires a database query per sentence. However, it could be sped up considerably using existing query optimization techniques for information retrieval. A second weakness is that performance of the LCA expansion currently requires sentence breaks. A modification of this approach would be to use a fixed-sized window rather than sentences as the atomic unit for expansion.

3.3.3 Discourse Based HMM Segmentation

The second method uses a Hidden Markov Model to model "marker words," or words which predict a topic change. The model consists of a state for the first sentence of a segment, one for the last sentence, and one for the remainder of the segment. So while the LCA segmenter relies on shifts in content, the HMM segmenter is relying on words which predict the beginning or end of a segment without regard to content. This is somewhat similar to the use of lexical features in [8]. The model is trained using segmented training data. For each state s, the probability of generating a word w is estimated as:

$$\frac{\#(w \wedge s)}{\#(s)}$$

In other words, the numbers of times word w is seen while in state s in the training data, over the total number of times in state s. In order to avoid over-fitting

the training data, words that were not seen were estimated using a backoff scheme as follows:

$$k \times p_{avg}(t)$$

where k is a small default weight (0.1 in these experiments) added to each state and $p_{avg}(t)$ is the average probability of term t across all of the states. The unseen words were given a fraction of this weight distributed according to the average probability of the word over all of the states. Words that did not appear at all in the training data received a default weight:

$$\frac{1.0}{0.5 \times |\text{training data}|}$$

This weight was used to estimate $p_{avg}(t)$ for non-occurring words. This essentially assumes that a non-occurring term occurred one half of one time in the training data. This prevents non-occuring terms from causing the probability to estimated as zero but does not allow them to have a large effect on the result. The resulting HMM is shown in Figure 3.5.

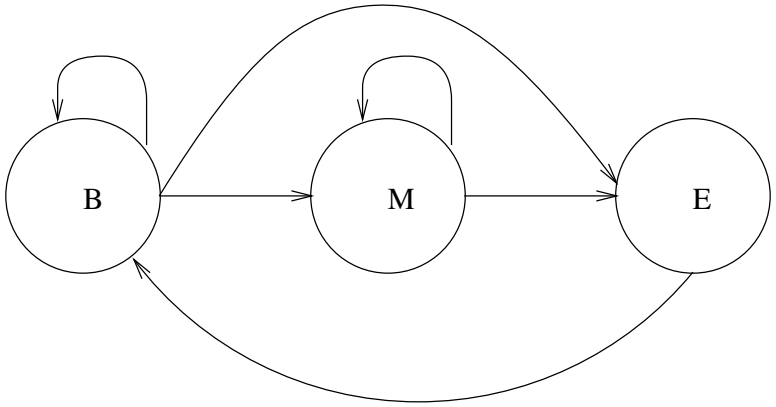

Figure 3.5. Discourse based HMM segmentation model.

The three states B, M and E are the beginning, middle, and end states, respectively. Each state has an output distribution determined from the training data as

described above. Each arrow in the diagram represents a transition probability, the A matrix in the standard formulation of HMMs given earlier. These probabilities are estimated from the counts of the state transitions in the segmented training data. Segmentation is performed using the Viterbi algorithm to determine the most likely state sequence that the model would go through to produce the sentences in the test set. A break is predicted each time the model enters state B.

3.3.3.1 Pros and Cons of the HMM method

One advantage of the HMM implementation is that it is very fast. Training time is approximately 15 minutes on the TDT training corpus and segmentation is extremely fast, as one would expect from an HMM with a small number of states. Also, unlike the LCA method, the HMM method can be used at the word level (although the current implementation works at the sentence level). The disadvantage of the HMM method is that it requires segmented training data.

3.3.4 Preliminary Results and Discussion

3.3.4.1 LCA Method

The LCA segmenter with the exponential length model achieves a 17.6% error probability as compared to the 30.6% error probability of the original LCA method described in the preliminary study that used only the offset heuristic. The new method is still heuristic in nature and a more principled use of the LCA concepts, as we shall see, improves performance further. One difficulty with the LCA method is that when one gives a query to LCA such as "Thank you and good-night." the concepts one gets back are essentially random. This difficulty will be circumvented by the combined method.

3.3.4.2 HMM Method

The HMM segmenter produces a segmentation with a 23% error rate on the TDT corpus. One caveat is that this approach may rely on the similarity of the training data to the test data somewhat heavily. Still, it shows that very simple discourse modeling can provide useful information. This method could be made more robust by explicitly modeling segues and other regularities of the source. For example, it would be more general to tag place names and names of reporters and to learn the probability of segment boundaries relative to the tags, rather than to the specific names, as the current approach does.

3.4 Combining the Two Methods

As pointed out, the two methods previously discussed make use of complementary features. Moreover, the two approaches have different strengths and weaknesses. A method will now be described that takes advantage of the strengths of the individual methods by means of a unified probabilistic model. This method achieves better performance than either method alone.

3.4.1 Sentence Clustering

The method of combining these two approaches is to use a Hidden Markov Model with explicit start and end of topic states, as in the previous HMM approach, and with multiple states representing topic models similar to those of Yamron *et al.* [3]. Recall that Yamron *et al.* clustered a hand segmented training corpus to produce one hundred language models. Instead of that, a similar clustering algorithm will be employed, but without segmented training data. Instead, the LCA concepts will be used to cluster the sentences. This allows language models to be produced without relying on hand segmented data. This is useful, because hand segmented data will generally

not be available for the production of topic specific language models, particularly in a dynamic news environment.

3.4.1.0.1 K-means Clustering The sentence clustering algorithm is a variation of k-means clustering. Recall that each sentence is represented by 100 concepts returned from the LCA expansion. The similarity of two sentences s_1 and s_2 is computed by the count of the common LCA concepts.

Each cluster is represented by a centroid vector containing the total counts of the LCA features for the member sentences. The similarity of a sentence s and a cluster c is computed using average link clustering as follows:

$$\frac{\sum_{s' \in c} sim(s, s')}{|c|}$$

where $|c|$ is the number of sentences in the cluster and $sim(s, s')$ is the similarity between sentences s and s' determined by coordination match of the LCA concepts for the two sentences.

The preliminary pass is done in random order. In order to produce N clusters, the first N sentences (drawn at random) are each placed into a new cluster. Each subsequent sentence is placed into the closest cluster. After the first pass is completed, ten additional passes are performed where each sentence is removed from its current cluster and reclustered. The resulting clusters are used to build unigram language models.

3.4.1.0.2 Language Model Construction Unigram language models are built from the original (pre-LCA) text of the sentences for each cluster. The probability of a word w for a cluster c is estimated as:

$$\frac{\#(w)}{\sum_{w' \in c} \#(w')}$$

In other words, estimated as the count of w normalized by the total number of word tokens in the original text of the sentences of cluster c. These unigram models are each treated as a state in a hidden Markov model similar to the approach in [3].

Unknown word probabilities for each state are estimated using the smoothing method previously described in Section 3.3.3. Each unknown word w is assigned the minimum within-state probability times the average probability of w across all of the states. The resulting distribution is re-normalized to insure that the probabilities sum to one.

3.4.1.0.3 Combined Model 100 language models were built using the above clustering approach. These models were used to replace the middle state in the HMM described earlier. The enhanced model is shown in Figure 3.6.

Once again, the states labeled B and E represent the states corresponding to the first and last sentence of a segment. The middle state has been replaced with the 100 language models generated from the clusters. The effect is to combine the marker word approach, used in the original HMM segmenter, with the content based features obtained by clustering the LCA concepts. The state transitions to each of the language model states are estimated using the middle state probability from the original HMM. The transition to each language model state receives $\frac{1}{100}$ of the transition probability to the middle state, i.e. each transition is considered equally likely. Segmentation is performed by means of the Viterbi algorithm, as before. The maximum likelihood state sequence is returned and a break is predicted each time the model passes through the beginning state.

The combined model was used to segment the TDT corpus and the direct evaluation was done as before. The result in the direct evaluation of the combined method was a 13% error probability, better than either method alone. However, the more important question is how this method performs in the context of the task of interest, and that will be measured using the indirect evaluation.

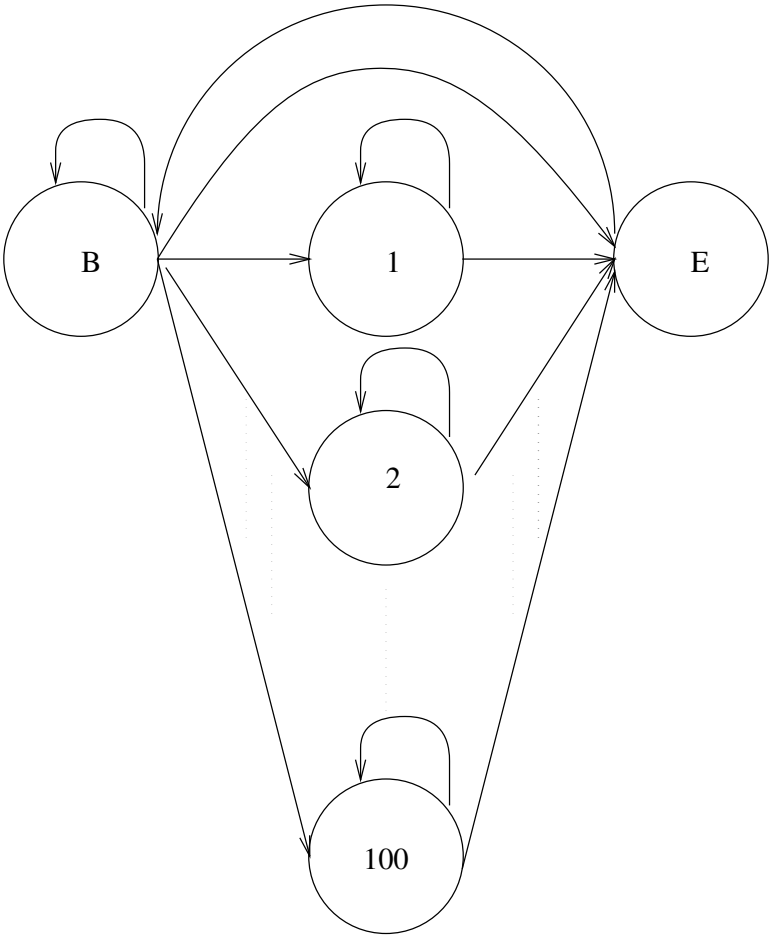

Figure 3.6. Combined discourse and content model.

3.5 Results of the Indirect Evaluation

Figure 3.7 shows DET curves for the new event detection task using the predicted breaks and the actual breaks. There is very little difference between the two curves particularly in the critical middle region of the curve. This indicates that segmentation in the context of new event detection is robust to the current error level. Further experimentation can be done to see if this will be case in general.

3.6 Discussion

The performance of the segmenter on the indirect evaluation shows that the new event detection task is robust in the face of a segmentation error rate of 13%. This is

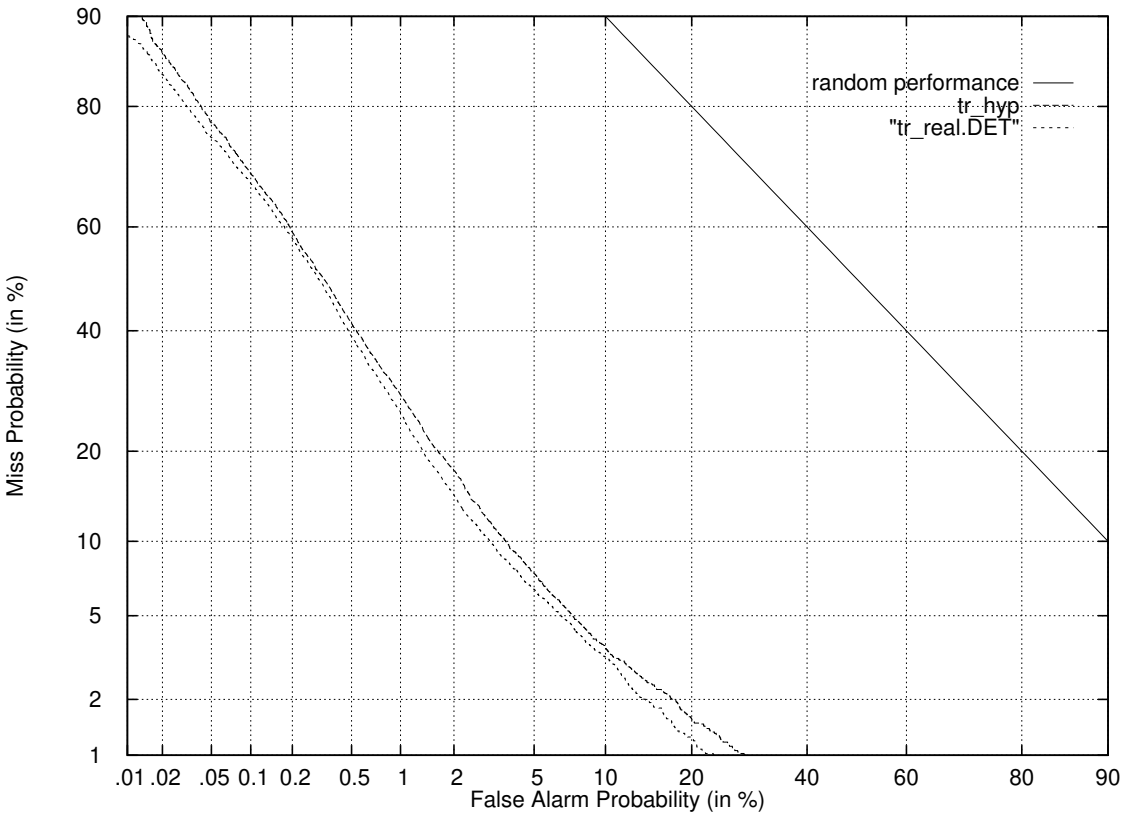

Figure 3.7. DET curve comparing predicted breaks to actual breaks.

consistent with results of using passage level evidence for interactive retrieval where it has been demonstrated that fixed sized passages are adequate for the improvement of retrieval effectiveness [1, 12]. Essentially, retrieval algorithms, and the derivative event detection algorithms, only require a coarse approximation of topic segmentation to perform well. For this reason, segmentation for the purposes of IR does not turn out to be a very interesting problem.

The remainder of the thesis focuses on what is, from an IR perspective, a more interesting problem, that of retrieval models. While segmentation for its own sake has not turned out to be as interesting as initially thought, the lessons learned about language modeling have been useful for the development of the new approach to retrieval discussed in the remainder of the thesis.

CHAPTER 4

LANGUAGE MODELS FOR INFORMATION RETRIEVAL

This chapter begins in Section 4.1 with an introduction to the new approach to retrieval that will be developed. Also discussed is the motivation for this approach including a discussion of document indexing models.

Section 4.2 covers Salton's recommendations for document indexing. Next, the differences between document and query text will be discussed in section 4.2 including the implications for probability estimation.

The description of the model takes place in Section 4.3. This includes the underlying idea and also the probability estimators used to implement it. After that, an empirical study is described in Section 4.4. The study compares the new model with an approach based on *tf.idf* weighting as used in other models of IR and then compares the baseline model with an enhanced model that uses a smoothing based estimator.

4.1 Chapter Introduction

This chapter discusses the development of a new model of text retrieval and shows its effectiveness empirically. The focus of this chapter is the ad hoc retrieval task, with additional aspects of retrieval covered in subsequent chapters. The underlying idea of this model in the context of ad hoc is quite simple. The retrieval task is carried out by estimating the probability of query production according to language models for each document. The intuition behind this approach is that users will have an idea of the documents of interest and will choose query terms that distinguish

these documents from the remainder of the collection. It is unknown what actually goes on in the mind of the user, so query production will be modeled as a random process. This is a departure from the usual approach to retrieval modeling described in Chapter 1 where the 2-Poisson model of index term assignment was discussed.

The early studies with this model showed that under certain circumstances, it was quite reasonable [28]. However, more recent studies on large heterogeneous document collections have shown that it does not apply in the more general case.

There are three reasons for this:

- Modern IR systems use a larger vocabulary.
- Modern IR systems rely on multiple term queries.
- Document length is highly variable in heterogeneous collections.

Regarding the first point, an indexing term assigned by the 2-Poisson model might be a specific scientific or medical term. For example, in a database of medical documents, the term "hemoglobin" might be assigned to a document and an expert reader would be able to determine whether or not the document was actually about hemoglobin. However, consider a database of newspaper articles and some terms from TREC topic number 202. Here is a list of terms from topic 202: "nuclear proliferation treaty violate monitor." What would it mean for a document to about the term "violate?" It is certainly not as clear as it is in the case of a word like "hemoglobin." In addition, it has been shown that in the case of a general vocabulary, words do not fit a mixture of two Poisson distributions very well [38]. So, even if one could reasonably claim that a document was or was not 'about' a single term, the model is just not correct in the case of a larger vocabulary.

Regarding the second point, in the above query, it is not any one word in the query that provides convincing evidence in favor of retrieving a document. Modern IR systems retrieve documents intended to be relevant to the aggregate of many

terms, not relevant to a single highly specific term. This is a second reason why the 2-Poisson model does not apply to the modern retrieval task.

Finally, the original studies with the 2-Poisson model assumed that documents were of relatively uniform length. This was a reasonable assumption for the small collections of abstracts used in the original studies but it does not apply to the large heterogeneous collections studied of late. These larger collections are more realistic in terms of text retrieval in a general setting. It is worth mentioning that the 2-Poisson model could be extended by means of normalized rates and doing so would circumvent this problem. However, in light of the previous two items, it is not likely that doing so would yield a useful model.

4.2 The Salton Approach to Indexing

An additional approach to indexing was developed by Salton [51]. Rather than using probabilistic models, Salton devised what he called a blueprint for automatic indexing, essentially a set of guidelines for the automatic indexing of documents.

Salton's prescription is summarized as follows:

1. Identify individual word tokens.
2. Delete common words (stopwords).
3. Strip suffixes using a stemmer.
4. Compute a weighting factor for each term.
5. Replace low-frequency terms with thesaurus classes.
6. Form phrases from high frequency terms.
7. Devise weights for the thesaurus classes and phrases.
8. Assign the appropriate words, thesaurus classes and phrases to each document.

Notice that these guidelines do not specify the weights for terms or phrases as would be done by a probabilistic model. In practice, a variation of *tf.idf* weighting would typically be used. Item 5 addresses a potential problem of rare terms. A very rare term may not be very useful for retrieval since it will not contribute to the score of very many documents. Consider a term that occurs exactly once in a collection, for example. In response to a user query, that term can cause the system to return at most one document. Item 5 suggests replacing such words with equivalence classes of related words. Likewise, item 6 suggests replacing common terms, which will tend to retrieve too many documents, by forming phrases from them.

One important point is that Salton's vector space model treats documents and queries as if they are fundamentally the same kind of object, vectors in T space where T is the number of unique terms. In this case, items 5 and 6 make sense because both prescribe ways of making better use of the available information in terms of discriminatory power of the features. However, these items are not typically done in practice. Part of the reason for this may be that queries and documents are really fundamentally different, and that is the view taken here. For simplicity, the case of single term features will be considered. Assume that queries consist of an unordered set of terms chosen by the user. The user will choose to add a term to the query if that term is likely to help find documents of interest. In this case, the query text is fundamentally different from the text of documents. It is text chosen by the user for the purpose of finding documents. Consider again the idea of forming phrases out of common terms. This would only help retrieval if the user can be relied upon to use phrases in queries in the same manner.

The language modeling approach will make the distinction between document text and query text. Document text is written natural language text. Query text is written with the express goal of finding a particular set of documents in which the

user is interested. This is a subtle but key distinction between the language modeling approach and the Kalt model [32].

4.2.1 The Kalt Model

Recall that the Kalt model treats documents as objects generated from a discrete memoryless channel. The source probabilities for a term t in a document d are estimated using the maximum likelihood estimate:

$$\hat{p}_{ml}(t|M_d) = \frac{tf_{(t,d)}}{dl_d}$$

where $tf_{(t,d)}$ is the raw term frequency of term t in document d and dl_d is the total number of tokens in document d.

4.3 The Language Modeling Approach

In the language modeling approach, the process of document generation is not modeled. It is the query generation process that is of interest for retrieval. The assumption is that the user has a prototypical document in mind and can, with a reasonable degree of accuracy, choose terms that will be likely to occur in that prototypical document and that will separate it from the remainder of the collection.

For this reason, a language model is inferred for each document and the documents are ranked according to the estimate of producing the query according to that model. When one makes the this distinction between documents and queries, it becomes clear why the maximum likelihood estimator is inadequate for the task. When one wishes to estimate the probability of a novel event from training data, the maximum likelihood estimator will over-fit when there is not enough data from which to estimate it.

In some sense, using Kalt's estimate for the *document* generation model is the best one can do. It is a perfect fit for the document. However, it may not be a perfect

fit for the *query*, because a single document is too small a sample to obtain a good estimate for the term probability.

The query generation probability $\hat{p}(Q|M_d)$, is the probability of producing the query given the language model of document d. To estimate this probability, the first step will be to estimate the probability for each term in the query. This probability will be estimated starting, as Kalt did, with the maximum likelihood estimate of the probability of term t in document d:

$$\hat{p}_{ml}(t|M_d) = \frac{tf_{(t,d)}}{dl_d}$$

where $tf_{(t,d)}$ is the raw term frequency of term t in document d and dl_d is the total number of tokens in document d. A simplifying assumption will be made. Assume that given a particular language model, the query terms occur independently. The maximum likelihood estimator gives rise to the ranking formula $\prod_{t \in Q} \hat{p}_{ml}(t, d)$ for each document.

4.3.1 Insufficient Data

There are two problems with the maximum likelihood estimator, or perhaps a better way to state this is that there are two symptoms of one underlying problem: insufficient data for the estimation of the maximum likelihood estimator. These two problems (or symptoms of the same problem) will now be examined a little more closely.

The obvious practical problem with this estimator is that we do not wish to assign a probability of zero for a document that is missing one or more of the query terms. Doing so would mean that if a user included several synonyms in the query, a document missing even one of them would not be retrieved

In addition to this practical consideration, from a probabilistic perspective, it is a somewhat radical assumption to infer that $p(t|M_d) = 0$, i.e., the fact that we have

not seen it does not make it impossible. Instead, we make the assumption that a non-occurring term is possible, but no more likely than what would be expected by chance in the collection, i.e., $\frac{cf_t}{cs}$, where cf_t is the raw count of term t in the collection and cs is the raw collection size or the total number of tokens in the collection. This provides us with a more reasonable distribution and circumvents the practical problem. It should be noted that in homogeneous databases, one may need to use a more careful smoothing estimate of the collection probability since, in some cases, the absence of a very frequently occurring word in a document (i.e., a word with the characteristics of a stopword) could conceivably contribute more to the score of that document in which it does not occur than it would for a document in which it does occur. This is not a problem in the collections studied, as they are heterogeneous in nature, and stopwords have been removed. However, this issue should be addressed in the future to insure that this approach will be immune to these pathological cases.

4.3.1.1 Small Sample Size

The other problem with this estimator was pointed out earlier. If we could get an arbitrarily large sample of data from M_d we could be reasonably confident in the maximum likelihood estimator. However we only have a document sized sample from that distribution and so the variation in the raw counts may partially be accounted for by randomness. In other words, if one were given two documents of equal length *generated from the same language model*, one would expect these two documents to have different numbers of occurrences of most of the terms due entirely to random variation.

4.3.2 Averaging

Since it is not possible to draw additional data from the same random process that generated each document, one can view the problem with the maximum likelihood estimator as one of insufficient data. To circumvent the problem of insufficient data,

we are going to need an estimate from a larger amount of data. That estimate is the mean probability estimate of t in documents containing it:

$$\hat{p}_{avg}(t) = \frac{\left(\sum_{d_{(t \in d)}} p_{ml}(t|M_d)\right)}{df_t}$$

where df_t is the document frequency of t. This is a more robust statistic in the sense that we have a lot more data from which to estimate it, but it too has a problem. It cannot be assumed that every document containing t is drawn from the same language model, and so there is some risk in using the mean to estimate $p(t|M_d)$. Furthermore, if the mean were used by itself, there would be no distinction between documents with different term frequencies.

In order to benefit from the robustness of this estimator, and to minimize the risk, the mean will be used to moderate the maximum likelihood estimator by combining the two estimates using the geometric distribution [69] as follows:

$$\hat{R}_{t,d} = \left(\frac{1.0}{(1.0 + \overline{f}_t)}\right) \times \left(\frac{\overline{f}_t}{(1.0 + \overline{f}_t)}\right)^{tf_{t,d}}$$

where $\overline{f}_t$ is the the mean term frequency of term t in documents where t occurs normalized by document length. The intuition behind this formula is that as the tf gets further away from the normalized mean, the mean probability becomes riskier to use as an estimate. For a somewhat related use of the geometric distribution see [23].

There are several reasons why the geometric function is a good choice in addition to the shape. In the first place, the mean of the distribution is equal to $\overline{f}_t$, the expected rate of occurrence according to the mean probability. Secondly, the variance of this distribution is larger than the mean. For this reason, the case where a large number of relatively low rates of occurrence along with relatively few cases where there is a large rate of occurrence are captured by this function. Finally, this function is

defined in terms of only the mean and the tf. This means that using this function does not cost very much in terms of time, either at indexing time or retrieval time, nor does it require additional space. IR systems need to be able to index Gigabytes of text at rates of hundreds of megabytes per hour and to process queries in seconds. Complex estimation techniques that require significant additional indexing time, even if effective, would not be acceptable for real systems. In addition, techniques that require significant additional space, such as storing an additional number per word occurrence, would not be feasible. This makes the geometric distribution an excellent choice from a systems engineering perspective.

4.3.3 Combining the Two Estimates

The geometric distribution will be used in the calculation of $\hat{p}(Q|M_d)$, the estimate of the probability of producing the query for a given document model as follows:

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \begin{cases} p_{ml}(t,d)^{(1.0-\hat{R}_{t,d})} \times p_{avg}(t)^{\hat{R}_{t,d}} & \text{if } tf_{(t,d)} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{cases}$$
$$\times \prod_{t \notin Q} 1.0 - \begin{cases} p_{ml}(t,d)^{(1.0-\hat{R}_{t,d})} \times p_{avg}(t)^{\hat{R}_{t,d}} & \text{if } tf_{(t,d)} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{cases}$$

In this formula, the first term is the probability of producing the terms in the query and the second term is the probability of not producing other terms. This can be regarded as a "background" model for each document that captures the likelihood of other terms for the document. This quantity essentially accounts for other terms that would be better discriminators of the document than the query terms, i.e., terms that would be unlikely to be left out of the query by users interested in the document. Also notice the risk function, $\hat{R}_{t,d}$ and the background probability $\frac{cf_t}{cs}$ mentioned earlier. This function is computed for each candidate document and the documents are ranked accordingly.

4.4 Empirical Results

4.4.1 Evaluation

Results are measured using the metrics of recall and precision. Table 4.1 shows the contingency table for retrieval of a set of documents where:

$$\begin{aligned} r &= \text{Relevant Set} \\ \overline{r} &= \text{Non-Relevant Set} \\ R &= \text{Retrieved Set} \\ \overline{R} &= \text{Non-Retrieved Set} \end{aligned}$$

Table 4.1. Contingency table for sets of documents.

$r \cap R$	$\overline{r} \cap R$
$r \cap \overline{R}$	$\overline{r} \cap \overline{R}$

If one were retrieving a set of documents, recall is the probability that a document has been retrieved given that it is a member of the set of relevant documents. Precision is the probability that a document is a member of the relevant set given that it has been retrieved. Defining these measures in terms of the contingency table yields:

$$\begin{aligned} \text{Recall} &= \frac{|r \cap R|}{|r|} = p(R|r) \\ \text{Precision} &= \frac{|r \cap R|}{|R|} = p(r|R) \end{aligned}$$

A third measure, fallout, is the probability that a document is not relevant given that is has been retrieved. Fallout, in terms of the contingency table looks like this:

$$\text{Fallout} = \frac{|\overline{r} \cap R|}{|\overline{r}|} = p(R|\overline{r})$$

Measuring performance in terms of recall and fallout would be more typical in a classification task. However, fallout is not a very interesting measure for retrieval.

Suppose, for example, that a collection consisted of one million documents, and suppose one hundred of them were relevant, which would be a typical proportion. In that case one can do a reasonable optimization for fallout by not retrieving any documents at all. Due to the huge disparity between the numbers of relevant and non-relevant documents, recall and precision are the preferred metrics.

For the evaluation of ranked retrieval, precision can be measured at several levels of recall to show the tradeoff. Other measures include the average precision over all relevant documents and precision after n documents have been retrieved for various values of n. Each of these measures will be reported for each of the experiments.

4.4.2 Data

Recall/precision experiments were performed on two data sets. The first set was TREC topics 202-250 on TREC disks 2 and 3, the TREC 4 ad hoc task, and the second was TREC topics 51-100 on TREC disk 3 using the concept fields. These query sets were chosen because they are quite different from each other. The 51-100 concept fields are essentially lists of good terms while topics 202-250 are 'natural language' queries consisting of one sentence each.

4.4.3 Implementation

A research prototype retrieval engine, known as Labrador, was implemented to test the language modeling approach. This engine was originally implemented as a high throughput retrieval system in the context of the topic segmentation work.

High throughput is achieved by keeping much (often all) of the index in memory to reduce the device wait times. In order to keep a large portion of the index in memory, it must be compressed. This is done using bit level index compression as described in [40]. The cost incurred for uncompressing the index is far less than the time for disk reads and so this is a very good tradeoff.

Labrador is best classified as a research prototype because it works very well but does not fail gracefully. The shortcomings of Labrador include lack of error checking and verification and a complete lack of documentation. This makes Labrador unsuitable for wider use in research and commercial environments, but it performed well for these experiments.

4.4.4 Recall/Precision Experiments

Recall/precision experiments were done to compare the language modeling approach against a baseline result. The baseline result was obtained using the INQUERY ranking formula, which uses Robertson's *tf* score, called $tfbel_{t,d}$ below, and a standard *idf* score. The function is defined as follows:

$$\begin{aligned} tfbel_{t,d} &= \frac{tf_{t,d}}{tf_{t,d} + 0.5 + 1.5\frac{len_d}{avgdoclen}} \\ idf_t &= \log(\frac{N+0.5}{df_t}) / \log(N+1) \end{aligned}$$

where $tf_{t,d}$ is the count of term t in document d, N is the number of documents in the collection, df_t is the number of documents containing term t, *avgdoclen* is the average document length in the entire collection and len_d is the length of document d.

Figure 4.1 shows the results for TREC topics 202-250 on TREC disks 2 and 3. The top line is the language modeling approach, notice the improvement over baseline. Also included are the UMass TREC 4 results. Notice that the results with the language modeling approach are competitive with the official UMass run even though the UMass run included additional features such as proximity pairs and unsupervised query expansion.

Table 4.2 shows the comparison of *tf.idf* to the language modeling approach in tabular form. In the table, we see the eleven point recall/precision results as well as

the non-interpolated average precision and precision figures for the top N documents for several values of N. The first two columns compare the baseline result to the language modeling approach.

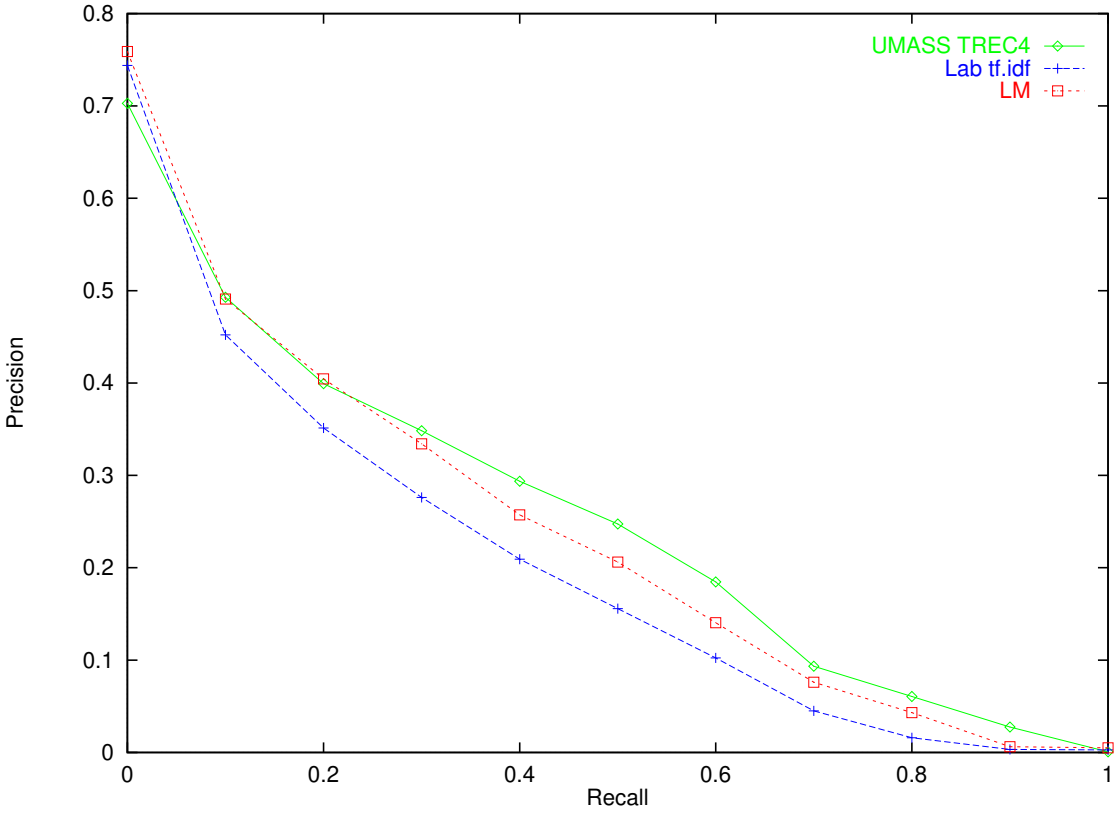

Figure 4.1. Comparison of *tf.idf* to the language modeling approach on TREC queries 202-250 on TREC disks 2 and 3.

The third column reports the percent change. Column four is of the form I/D where I is the count of queries for which performance improved using the new method and D is count of queries for which performance was different. Column five reports significance values according to the sign test and column six does likewise according to the Wilcoxon test. The entries in these two columns marked with a star indicate a statistically significant difference at the 0.05 level. Note that these are one sided tests.

Notice that on the eleven point recall/precision section, the language modeling approach achieves better precision at all levels of recall, significantly at several levels.

Table 4.2. Comparison of *tf.idf* to the language modeling approach on TREC queries 202-250 on TREC disks 2 and 3.

	tf.idf	LModel	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3201	3364	+5.09	36/43	0.0000⋆	0.0002⋆
Precision						
at 0.00	0.7439	0.7590	+2.0	10/22	0.7383	0.5709
at 0.10	0.4521	0.4910	+8.6	24/42	0.2204	0.0761
at 0.20	0.3514	0.4045	+15.1	27/44	0.0871	0.0081⋆
at 0.30	0.2761	0.3342	+21.0	28/43	0.0330⋆	0.0054⋆
at 0.40	0.2093	0.2572	+22.9	25/39	0.0541	0.0158⋆
at 0.50	0.1558	0.2061	+32.3	24/35	0.0205⋆	0.0018⋆
at 0.60	0.1024	0.1405	+37.1	22/27	0.0008⋆	0.0027⋆
at 0.70	0.0451	0.0760	+68.7	13/15	0.0037⋆	0.0062⋆
at 0.80	0.0160	0.0432	+169.6	9/10	0.0107⋆	0.0035⋆
at 0.90	0.0033	0.0063	+89.3	2/3	0.5000	undef
at 1.00	0.0028	0.0050	+76.9	2/3	0.5000	undef
Avg:	0.1868	0.2233	+19.55	32/49	0.0222⋆	0.0003⋆
Precision at:						
5 docs:	0.4939	0.5020	+1.7	10/21	0.6682	0.4106
10 docs:	0.4449	0.4898	+10.1	22/30	0.0081⋆	0.0154⋆
15 docs:	0.3932	0.4435	+12.8	19/26	0.0145⋆	0.0038⋆
20 docs:	0.3643	0.4051	+11.2	22/34	0.0607	0.0218⋆
30 docs:	0.3313	0.3707	+11.9	28/41	0.0138⋆	0.0070⋆
100 docs:	0.2157	0.2500	+15.9	32/42	0.0005⋆	0.0003⋆
200 docs:	0.1655	0.1903	+15.0	35/44	0.0001⋆	0.0000⋆
500 docs:	0.1004	0.1119	+11.4	36/44	0.0000⋆	0.0000⋆
1000 docs:	0.0653	0.0687	+5.1	36/43	0.0000⋆	0.0002⋆
R-Precision:	0.2473	0.2876	+16.32	34/43	0.0001⋆	0.0000⋆

Also notice that there is a significant improvement in recall, uninterpolated average precision and R-precision, the precision after R documents where R is equal to the number of relevant documents for each query. On the second part of the figure, there is again an improvement at all levels of recall, many of them statistically significant. It should be pointed out that using the only the maximum likelihood estimator yields precision results worse at all levels of recall and worse by approximately 26% on average.

Figure 4.2 shows the results for TREC topics 51-100 on TREC disk 3. Once again the language modeling approach shows improvement over the baseline. See Table 4.3 for the same data in tabular form. Improvement in precision can be seen at most levels of recall on the eleven point chart. The results on these two data sets indicate

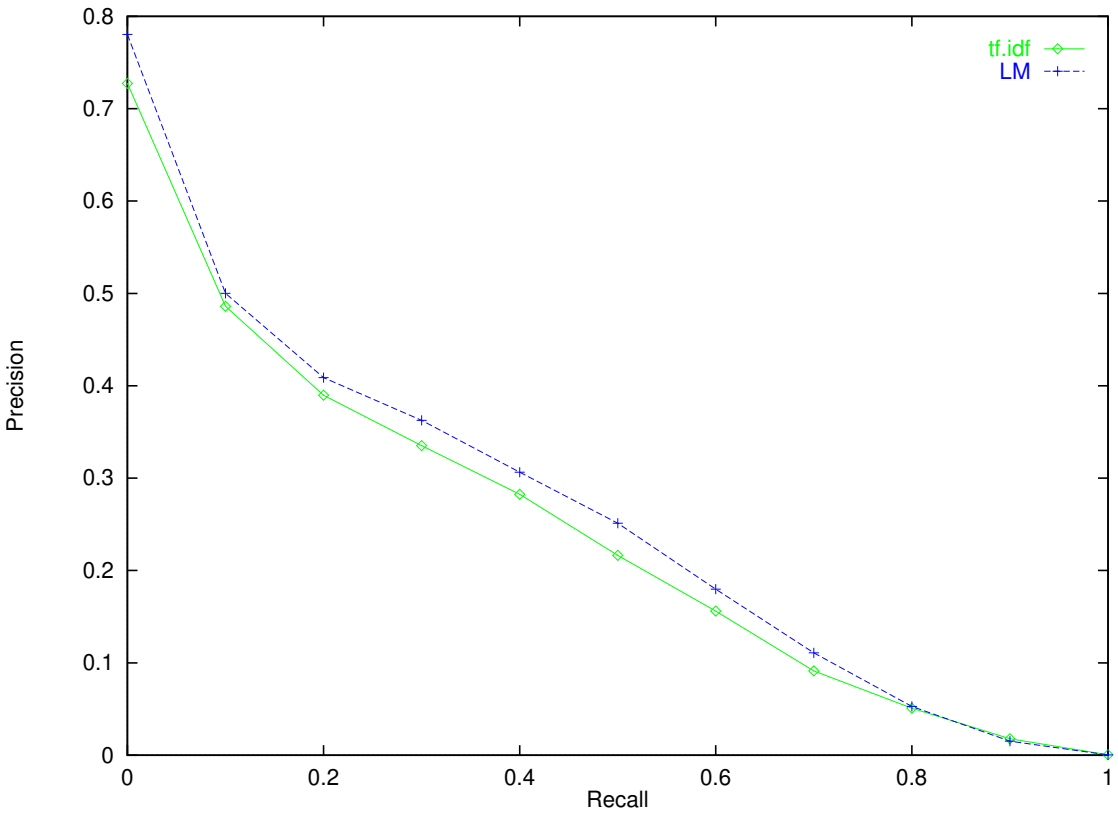

Figure 4.2. Comparison of *tf.idf* to the language modeling approach on TREC queries 51-100 on TREC disk 3.

that the language modeling approach is a good alternative to the more traditional *tf.idf* approach to retrieval. The next question is how the model can be used as a guide for the improvement of retrieval effectiveness.

4.4.5 Improving the Basic Model

The language modeling approach depends on good estimation. In order to improve retrieval effectiveness, one should improve the estimate of the query generation probabilities. A simple improvement of the estimate developed in Section 4.3 is to smooth the estimates of the average probability for terms with low document frequency. The estimate of the average probability of these terms is based on a small amount of data and so could be sensitive to outliers.

Table 4.3. Comparison of *tf.idf* to the language modeling approach on TREC queries 51-100 on TREC disk 3.

	tf.idf	LModel	% chng.	I/D	Sign	Wilcoxon
Relevant:	10485	10485				
Rel. ret.:	5818	6105	+4.93	32/42	0.0005⋆	0.0003⋆
Precision						
at 0.00	0.7274	0.7805	+7.3	10/22	0.7383	0.2961
at 0.10	0.4861	0.5002	+2.9	26/44	0.1456	0.1017
at 0.20	0.3898	0.4088	+4.9	24/45	0.3830	0.1405
at 0.30	0.3352	0.3626	+8.2	28/47	0.1215	0.0277⋆
at 0.40	0.2826	0.3064	+8.4	25/45	0.2757	0.0286⋆
at 0.50	0.2163	0.2512	+16.2	26/40	0.0403⋆	0.0007⋆
at 0.60	0.1561	0.1798	+15.2	20/30	0.0494⋆	0.0025⋆
at 0.70	0.0913	0.1109	+21.5	14/22	0.1431	0.0288⋆
at 0.80	0.0510	0.0529	+3.7	8/13	0.2905	0.2108
at 0.90	0.0179	0.0152	−14.9	1/4	0.3125	undef
at 1.00	0.0005	0.0004	−11.9	1/2	0.7500	undef
Avg:	0.2286	0.2486	+8.74	32/50	0.0325⋆	0.0015⋆
Precision at:						
5 docs:	0.5320	0.5960	+12.0	15/21	0.0392⋆	0.0125⋆
10 docs:	0.5080	0.5260	+3.5	14/30	0.7077	0.1938
15 docs:	0.4933	0.5053	+2.4	14/28	0.5747	0.3002
20 docs:	0.4670	0.4890	+4.7	16/34	0.6962	0.1260
30 docs:	0.4293	0.4593	+7.0	20/32	0.1077	0.0095⋆
100 docs:	0.3344	0.3562	+6.5	29/45	0.0362⋆	0.0076⋆
200 docs:	0.2670	0.2852	+6.8	29/44	0.0244⋆	0.0009⋆
500 docs:	0.1797	0.1881	+4.7	30/42	0.0040⋆	0.0011⋆
1000 docs:	0.1164	0.1221	+4.9	32/42	0.0005⋆	0.0003⋆
R-Precision:	0.2836	0.3013	+6.24	30/43	0.0069⋆	0.0052⋆

In order to correct for this, the low document frequency data (using a cutoff of $df = 100$) was binned by document frequency, and the binned estimate was used for the average. This new estimate of the average is incorporated into the ranking formula, as before, and rerun on TREC queries 202-250 against TREC disks 2 and 3. The results are shown in Figure 4.3. A small improvement over baseline can be seen from the graph. For an alternative view of these results see Table 4.4. The results show a statistically significant improvement in precision at several levels of recall. The average precision is also improved.

Running the new model on the second query set, TREC queries 51-100 against TREC disk 3, yields the result shown in figure 4.4. The results have improved, but so slightly, it is difficult see from the graph. However, when these results are viewed in

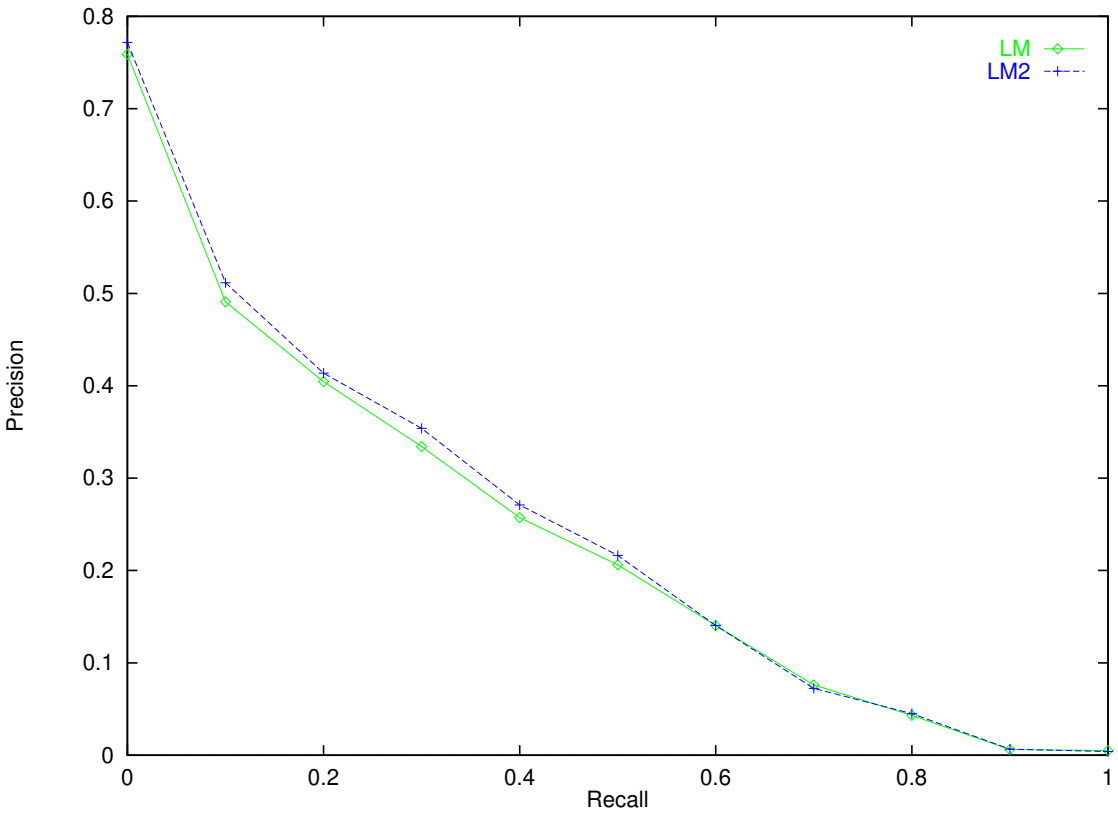

Figure 4.3. Comparison of the original language modeling approach to the new language modeling approach on TREC queries 202-250 on TREC disks 2 and 3.

tabular form (see Table 4.5), it can be seen that many of the improvements, though modest, are statistically significant.

Note that the average query length for topics 51-100 is considerably longer than for the previous set. A reasonable conjecture is that that smaller improvement on this query set is due to the longer average query length. It appears that for low frequency terms, the effects on the average due to outliers is just as likely to cause an overestimate, as it is to cause underestimate, and so these effects cancel each other out when there are more terms in the query. However, this is only a conjecture, the verification of which is left for future work. The main point to take away from the smoothing experiments is that improving the probability estimation, in this case by a simple smoothing technique, improves retrieval performance. This is evidence further evidence that the language modeling approach is a reasonable approach to retrieval.

Table 4.4. Comparison of the original language modeling approach to the new language modeling approach on TREC queries 202-250 on TREC disks 2 and 3.

	LModel	LModel2	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3364	3350	−0.42	16/33	0.5000	0.4432
Precision						
at 0.00	0.7590	0.7717	+1.7	11/17	0.1662	0.1137
at 0.10	0.4910	0.5115	+4.2	25/41	0.1055	0.0194⋆
at 0.20	0.4045	0.4137	+2.3	23/42	0.3220	0.2100
at 0.30	0.3342	0.3539	+5.9	26/42	0.0821	0.0275⋆
at 0.40	0.2572	0.2709	+5.3	23/37	0.0939	0.0420⋆
at 0.50	0.2061	0.2164	+5.0	23/33	0.0175⋆	0.0222⋆
at 0.60	0.1405	0.1405	−0.0	15/24	0.9242	0.8197
at 0.70	0.0760	0.0724	−4.8	4/14	0.0898	0.0886
at 0.80	0.0432	0.0450	+4.1	5/9	0.5000	undef
at 0.90	0.0063	0.0065	+4.6	2/3	0.5000	undef
at 1.00	0.0050	0.0040	−19.1	2/3	0.8750	undef
Avg:	0.2233	0.2318	+3.81	34/49	0.0047⋆	0.0055⋆
Precision at:						
5 docs:	0.5020	0.5469	+8.9	13/17	0.0245⋆	0.0176⋆
10 docs:	0.4898	0.5082	+3.7	12/22	0.4159	0.1532
15 docs:	0.4435	0.4571	+3.1	14/23	0.2024	0.1007
20 docs:	0.4051	0.4235	+4.5	18/25	0.0216⋆	0.0083⋆
30 docs:	0.3707	0.3755	+1.3	16/34	0.6962	0.3222
100 docs:	0.2500	0.2655	+6.2	28/39	0.0047⋆	0.0005⋆
200 docs:	0.1903	0.1932	+1.5	18/30	0.1808	0.1226
500 docs:	0.1119	0.1128	+0.8	21/37	0.2557	0.1615
1000 docs:	0.0687	0.0684	−0.4	16/33	0.5000	0.4432
R-Precision:	0.2876	0.2928	+1.79	19/34	0.3038	0.1485

4.5 Discussion

This chapter introduced the language modeling approach to text retrieval. A key component of this approach is the estimation of the document model probabilities. The smoothing experiments show that improving estimation of the probabilities leads to improved retrieval. This helps to provide proof of concept of the language modeling approach since improved estimation has the effect that the model predicts.

Compared to the vector space model, the language modeling approach shares the virtue of simplicity, but is different in the following ways:

1. Queries and documents are treated as fundamentally different objects.

2. The semantics of the ranking function are an integral part of the model.

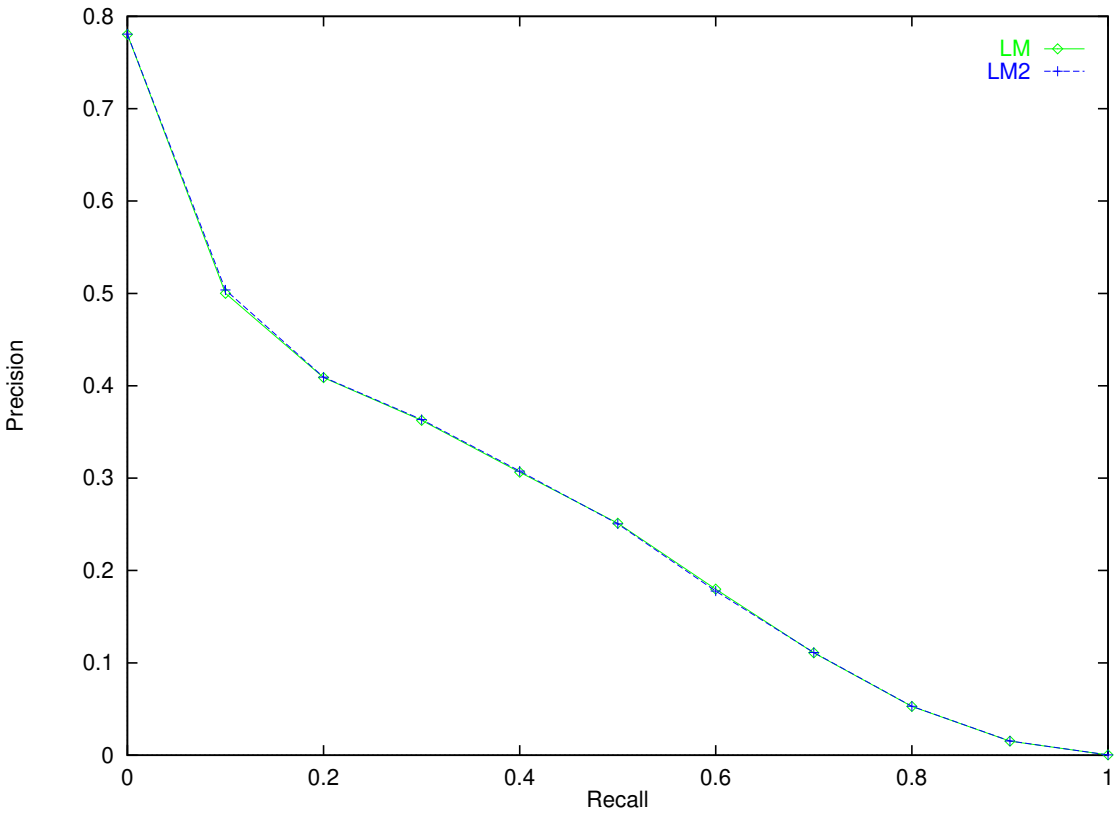

Figure 4.4. Comparison of the original language modeling approach to the new language modeling approach on TREC queries 51-100 on TREC disk 3.

3. The model predicts methods of improving retrieval effectiveness.

Recall that in the vector space model, queries and documents are treated as objects in the same space. However, the language modeling approach makes a distinction between them. When one makes this distinction, the problem of estimation becomes clearer. The document level statistics can be used to estimate a language model from which the query generation probabilities, according to that model, can be calculated. This requires that the estimates be robust to the possible variations in the rate of occurrence of the query terms.

The contrasting view, where the estimation problem is viewed as the *document* generation probability, could lead one to rely on the maximum likelihood estimator exclusively, since it is essentially a perfect fit to each document.

Table 4.5. Comparison of the original language modeling approach to the new language modeling approach on TREC queries 51-100 on TREC disk 3.

	LModel	LModel2	% chng.	I/D	Sign	Wilcoxon
Relevant:	10485	10485				
Rel. ret.:	6105	6107	+0.03	3/5	0.5000	undef
Precision						
at 0.00	0.7805	0.7807	+0.0	3/5	0.5000	undef
at 0.10	0.5002	0.5038	+0.7	16/20	0.0059⋆	0.0020⋆
at 0.20	0.4088	0.4093	+0.1	16/25	0.1148	0.0959
at 0.30	0.3626	0.3634	+0.2	13/18	0.0481⋆	0.0238⋆
at 0.40	0.3064	0.3077	+0.4	16/24	0.0758	0.0198⋆
at 0.50	0.2512	0.2505	−0.3	11/25	0.3450	0.2059
at 0.60	0.1798	0.1777	−1.2	10/20	0.5881	0.3826
at 0.70	0.1109	0.1113	+0.3	9/12	0.0730	0.0356⋆
at 0.80	0.0529	0.0530	+0.1	5/8	0.3633	undef
at 0.90	0.0152	0.0154	+0.9	1/2	0.7500	undef
at 1.00	0.0004	0.0004	+0.4	1/1	0.5000	undef
Avg:	0.2486	0.2488	+0.08	28/39	0.0047⋆	0.0254⋆
Precision at:						
5 docs:	0.5960	0.6000	+0.7	1/1	0.5000	undef
10 docs:	0.5260	0.5260	+0.0	0/0	1.0000	undef
15 docs:	0.5053	0.5093	+0.8	3/3	0.1250	undef
20 docs:	0.4890	0.4920	+0.6	4/5	0.1875	undef
30 docs:	0.4593	0.4613	+0.4	5/8	0.3633	undef
100 docs:	0.3562	0.3568	+0.2	5/7	0.2266	undef
200 docs:	0.2852	0.2859	+0.2	8/11	0.1133	0.0548
500 docs:	0.1881	0.1884	+0.1	6/9	0.2539	undef
1000 docs:	0.1221	0.1221	+0.0	3/5	0.5000	undef
R-Precision:	0.3013	0.3011	−0.08	8/12	0.9270	0.7349

Compared to more standard models of probabilistic retrieval, the language modeling approach shares the desirable property of explaining the semantics of the numbers. The major difference is that in order to estimate indexing term assignment probabilities, the traditional models tend to use heuristic techniques to estimate the probability that a given document describes a particular concept. This make these model more complex that the language modeling approach which is conceptually very simple. The value of this simplicity will be demonstrated in Chapter 5 where query expansion techniques will be developed that are fairly obvious extensions to the language modeling approach.

CHAPTER 5

QUERY EXPANSION TECHNIQUES

This chapter begins with an introduction to query expansion techniques and their role in retrieval systems. Next, the technique of relevance feedback is developed using the language modeling approach in section 5.2. Empirical results comparing the approach to an existing relevance feedback technique are also presented. A similar technique is applied to document routing and empirical results are presented and discussed.

Following that, is a discussion of techniques for query expansion without user relevance judgments in Section 5.3. Several techniques are discussed, and two techniques developed using the language modeling approach are described along with experimental results.

5.1 Chapter Introduction

The ranking formula developed in Chapter 4 is only part of a complete retrieval model. Information retrieval researchers have developed several techniques that improve results considerably. For example, the INQUERY inference network model supports a rich class of query operators including Boolean operators, proximity constraints, phrases, and passage level evidence. In addition, query expansion techniques, such as relevance feedback and local context analysis, have been shown empirically to be effective, and are an integral part of the INQUERY system. This chapter focuses on query expansion techniques. The other components of retrieval systems will be addressed in chapter 6.

5.1.1 Term Mismatch

Why query expansion? The fundamental notion of the language modeling approach to retrieval is that users can choose query terms that will be likely to appear in documents of interest, and less likely to occur in the collection as a whole. However, often there will be some vocabulary mismatch between a query and a subset of the documents of interest. For example, a user interested in documents about cancer therapies may enter the query: 'cancer chemotherapy,' which contains two good terms. However, documents that described 'tumors' or 'malignancies' treated with 'radiation' would be missed. IR systems can remedy this problem in two ways.

The first is relevance feedback. The query is first run as usual. The user then marks some of the retrieved documents as relevant. The system then uses these judgments to improve the initial query, usually by adding some features from the relevant documents and possibly by re-weighting the initial query terms.

The second technique is to modify the query without relevance information. Features can be selected according to the degree of co-occurrence with the initial query terms. Several different techniques have been developed for co-occurrence based query expansion and several of them will be discussed. For a more detailed discussion of query expansion techniques see [67].

5.1.2 Query Expansion in the Language Modeling Approach

Query expansion techniques have a natural interpretation in the language modeling approach. Recall that the underlying assumption of this approach is that users can choose query terms that are likely to occur in documents in which they would be interested, and that separate these documents from the ones in which they are not interested. Also note that this notion has been developed into a ranking formula by means of probabilistic language models. Since probabilities can be estimated as

shown in Chapter 4 to, in some sense, evaluate terms chosen by the user, the same probabilities can be used by the system to 'choose' terms in much the same way.

5.1.2.1 Interactive Retrieval with Relevance Feedback

Consider the relevance feedback case, where a small number of documents are judged relevant by the user. The relevance of all of the remaining documents is unknown to the system. The system can estimate the probability of producing terms from each document's probability distribution. From this, a random sample could be drawn from that distribution, but this would be undesirable since, of course, the most common terms would tend to be drawn.

However, it is also possible to estimate the probability of each term in the collection as a whole using the term counts. The log ratio of the two can then be used to do as the user does: choose terms likely in documents of interest but unlikely in the collection as a whole. The details of this process are discussed in Section 5.2

5.1.2.2 Document Routing

In the routing case, a training collection is available with a large number of relevance judgments, both positive and negative, for a particular query. The task is to use the training collection to construct a query to be used on a second collection for which relevance information is not available.

Since both relevant and non-relevant documents are known, the ratio method can utilize this additional information by estimating probabilities for both sets. This process is described in more detail in section 5.2.2. Once again, the task is to choose terms associated with documents of interest and to avoid those associated with other documents. The ratio method provides a simple mechanism to do that.

5.2 Relevance Feedback

As mentioned, modern retrieval systems can provide improved retrieval effectiveness by the process of relevance feedback. In an interactive setting, the user poses an initial query and then marks several documents in the resulting ranked list as relevant to the query. The system then chooses features from these relevant documents to add to the query based on the collection statistics and reruns the modified query.

5.2.1 The Ratio Method

Incorporating relevance feedback into the INQUERY model was a very difficult problem and the subject of a PhD dissertation by Haines [17]. The solution developed by Haines was to enhance the inference network model with a new class of nodes called annotation nodes. For each query term, a structure consisting of three new nodes, along with associated inferences, was added to the original query network. The resulting network can be calculated in an efficient manner as shown by Haines [17]. The only drawback with this formulation is that is quite complex.

Incorporating relevance feedback into the language modeling approach is considerably easier. Recall that in the language modeling approach, the prototypical user considers the documents of interest and chooses terms that will separate these documents from the rest of the collection. An analogous method will be used to choose terms for relevance feedback. The judged documents can be used to estimate a language model of the documents of interest using the method described in Chapter 4. The probability of a term t in the collection as a whole can be estimated, as before, by $\frac{cf_t}{cs}$, the collection frequency of t normalized by the total number of tokens in the collection.

These two probability models can then be used in manner similar to that used by Beeferman *et al.* [8] for the purposes of topic boundary prediction. Recall from

section 2.3.4 that the method they used was to take the log ratio of a long range language model vs. a short range language model in order to predict topic shifts.

In the relevance feedback case, the two estimates from above will be used in a similar manner to predict useful terms. Terms can then be ranked according to the probability of occurrence according to the relevant document models as compared to the collection as a whole, i.e. terms can be ranked according to the sum of the log ratio of each relevant model vs. the collection model.

$$L_t = \sum_{d \in R} \log \left(\frac{P(t|M_d)}{\frac{cf_t}{cs}} \right)$$

where R is the set of relevant documents, $P(t|M_d)$ is the probability of term t given the document model for d as defined in Chapter 4, cf_t is the raw count of term t in the collection and cs is the raw collection size. Terms are ranked according to this ratio and the top N are added to the initial query.

The results of feeding back one relevant document and adding five terms to the initial query are shown in Figure 5.1. The baseline result was obtained using Rocchio weighting [49] as described in Section 2.2.7.2. The language modeling approach is slightly better at the top of the ranking but Rocchio is better at most of the points.

Also see Table 5.1. Notice the increased precision at the top of the ranking for both techniques over the baseline reported in Table 4.2. Overall, the two techniques tend to track each other reasonably well, though the Rocchio method outperforms the language modeling approach at most levels of recall and on average.

The results of feeding back two relevant documents and adding five terms to the initial query are shown in Figure 5.2 Once again, the baseline technique is Rocchio. With two documents, the drop off in average performance that occurred with a single document no longer occurs.

The alternative view of these results is shown in Table 5.2. Notice that the improvements of the language modeling approach over Rocchio are not statistically

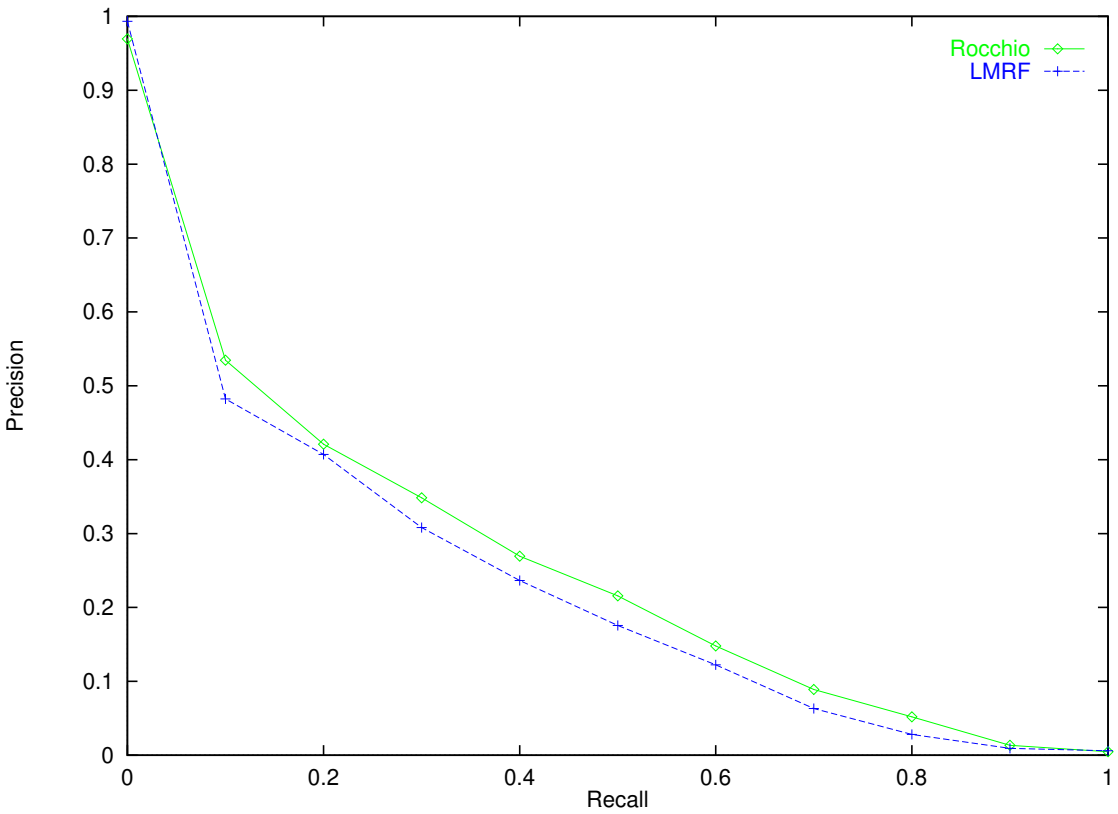

Figure 5.1. Comparison of Rocchio to the Language modeling approach using 1 document and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

significant. Again, as with a single document, the two techniques provide similar performance at most levels of recall.

Figure 5.3 and figure 5.4 both show results of feeding back ten relevant documents adding five and ten terms respectively. The tabular views are also included as tables 5.3 and 5.4. As in the previous experiments, the baseline technique is Rocchio.

Note that for five terms, recall, average precision and precision at several levels of recall are significantly better using the language modeling approach. However, it should be noted that ten relevant documents is a large number to expect from a typical user. The results for ten terms are similar. Also note that both techniques show improved results by the addition of more terms, as expected.

The main point to take away from this series of experiments is that a relevance feedback technique that follows directly from the language modeling approach works

Table 5.1. Comparison of Rocchio to the Language modeling approach using 1 document and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

	Rocchio	LMRF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3366	3270	−2.85	16/39	0.1684	0.3004
Precision						
at 0.00	0.9694	0.9932	+2.5	2/2	0.2500	undef
at 0.10	0.5346	0.4822	−9.8	21/44	0.4402	0.0707
at 0.20	0.4209	0.4070	−3.3	22/44	0.5598	0.2997
at 0.30	0.3484	0.3082	−11.5	19/43	0.2712	0.1413
at 0.40	0.2694	0.2365	−12.2	15/40	0.0769	0.1323
at 0.50	0.2156	0.1757	−18.5	12/36	0.0326⋆	0.0434⋆
at 0.60	0.1478	0.1224	−17.2	9/29	0.0307⋆	0.0355⋆
at 0.70	0.0890	0.0633	−28.8	7/22	0.0669	0.0412⋆
at 0.80	0.0519	0.0281	−45.9	4/12	0.1938	0.0680
at 0.90	0.0135	0.0094	−29.9	2/6	0.3438	undef
at 1.00	0.0047	0.0060	+26.4	2/3	0.5000	undef
Avg:	0.2410	0.2147	−10.91	22/49	0.2841	0.0841
Precision at:						
5 docs:	0.5878	0.5551	−5.6	13/30	0.2923	0.3036
10 docs:	0.5102	0.4776	−6.4	15/35	0.2498	0.2041
15 docs:	0.4803	0.4367	−9.1	15/37	0.1620	0.0861
20 docs:	0.4490	0.4102	−8.6	13/37	0.0494⋆	0.0873
30 docs:	0.3891	0.3667	−5.8	21/43	0.5000	0.2024
100 docs:	0.2537	0.2327	−8.3	19/43	0.2712	0.0961
200 docs:	0.1835	0.1788	−2.6	20/42	0.4388	0.3399
500 docs:	0.1119	0.1070	−4.4	19/42	0.3220	0.1976
1000 docs:	0.0687	0.0667	−2.9	16/39	0.1684	0.3004
R-Precision:	0.2930	0.2647	−9.66	18/43	0.1802	0.0626

well without any *ad hoc* additions. This is further proof of concept of the language modeling approach to IR.

Note that in most IR systems, relevance feedback techniques make use of term weighting. No attempt was made to use term weighting in these experiments. This matter is discussed in more detail in section 6.4.

5.2.2 Information Routing

Recall that, in the routing task, a training collection with relevance judgments, positive and negative, is provided. The task is to develop a query that will perform well on a test set of new documents. The language modeling approach to informa-

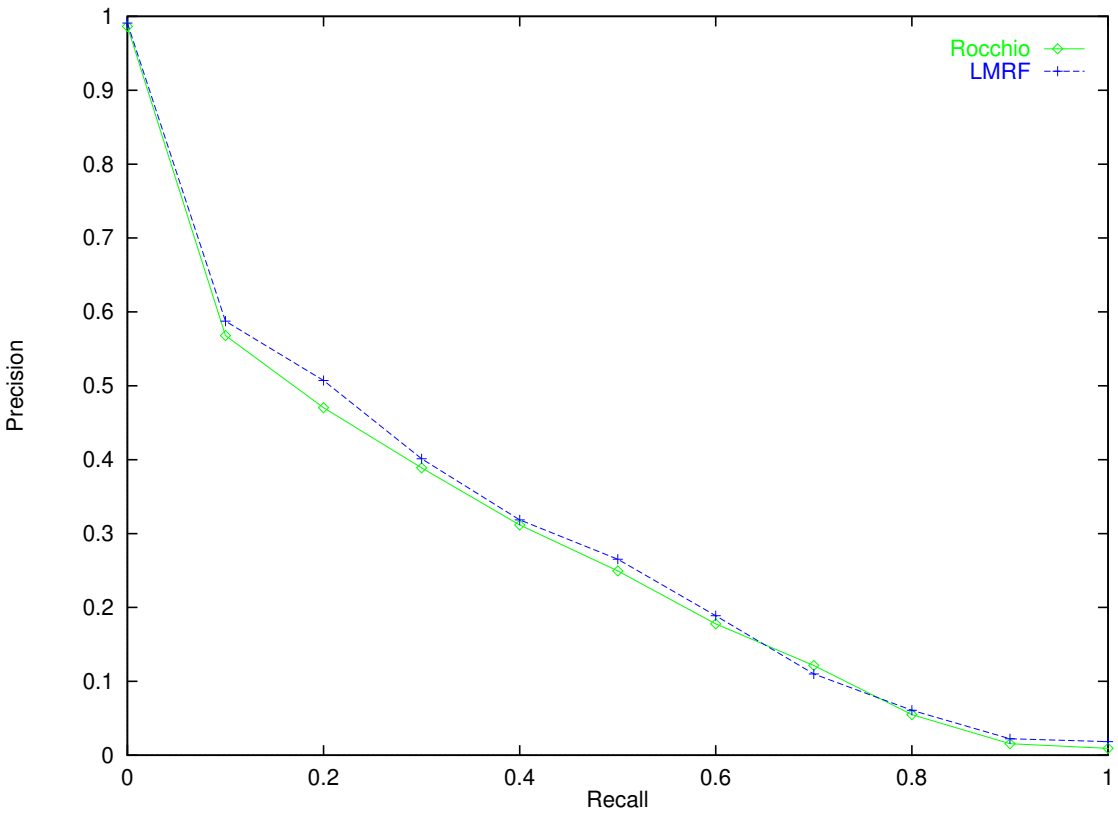

Figure 5.2. Comparison of Rocchio to the Language modeling approach using 2 documents and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

tion routing will use similar techniques to those used for relevance feedback. The experiments described will show two different ratio methods. The first is identical to the relevance feedback method described in section 5.2 where documents were either relevant or unknown. The second method uses models of both the relevant and non-relevant sets of documents. It will be shown that the negative information is very useful in the context of the language modeling approach, just as it is in existing approaches to routing.

5.2.2.1 Ratio Methods With More Data

In Section 5.2, the ratio method of relevance feedback was described. In this section, the ratio method will be described in the context of the routing task. In addition, a second ratio method will be developed to make use of the additional

Table 5.2. Comparison of Rocchio to the Language modeling approach using 2 documents and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

	Rocchio	LMRF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3650	3590	−1.64	24/44	0.7743	0.4351
Precision						
at 0.00	0.9864	0.9908	+0.4	2/3	0.5000	undef
at 0.10	0.5679	0.5874	+3.4	22/45	0.6170	0.2882
at 0.20	0.4703	0.5072	+7.8	26/46	0.2307	0.1361
at 0.30	0.3889	0.4013	+3.2	26/45	0.1856	0.2473
at 0.40	0.3115	0.3186	+2.3	23/40	0.2148	0.2726
at 0.50	0.2494	0.2652	+6.4	22/39	0.2612	0.2173
at 0.60	0.1778	0.1888	+6.2	17/33	0.5000	0.3116
at 0.70	0.1215	0.1099	−9.6	7/25	0.0216⋆	0.1264
at 0.80	0.0551	0.0608	+10.4	8/15	0.5000	0.4548
at 0.90	0.0156	0.0222	+42.2	4/6	0.3438	undef
at 1.00	0.0093	0.0184	+98.6	2/3	0.5000	undef
Avg:	0.2739	0.2825	+3.16	27/49	0.2841	0.2003
Precision at:						
5 docs:	0.6735	0.7061	+4.8	21/36	0.2025	0.1853
10 docs:	0.5939	0.6102	+2.7	24/43	0.2712	0.2710
15 docs:	0.5388	0.5537	+2.8	21/40	0.4373	0.2218
20 docs:	0.5092	0.4939	−3.0	18/41	0.2664	0.4053
30 docs:	0.4456	0.4367	−2.0	19/44	0.2257	0.4744
100 docs:	0.2890	0.2853	−1.3	25/46	0.7693	0.5737
200 docs:	0.2067	0.2120	+2.6	27/43	0.0631	0.1509
500 docs:	0.1225	0.1200	−2.1	24/42	0.8600	0.4502
1000 docs:	0.0745	0.0733	−1.6	24/44	0.7743	0.4351
R-Precision:	0.3238	0.3264	+0.79	25/45	0.2757	0.3258

information available in the routing task. Empirical results will be shown in section 5.2.2.2.

As in the relevance feedback case, terms are ranked by the log ratio of the probability in the judged relevant set vs. the collection as a whole. These two term sets will be used to construct queries as described below. In the results section, this method will be referred to as ratio 1.

The second ratio method, referred to as ratio 2 in the results section, uses the log ratio of the average probability in judged relevant documents vs. the average probability in judged non-relevant documents as shown.

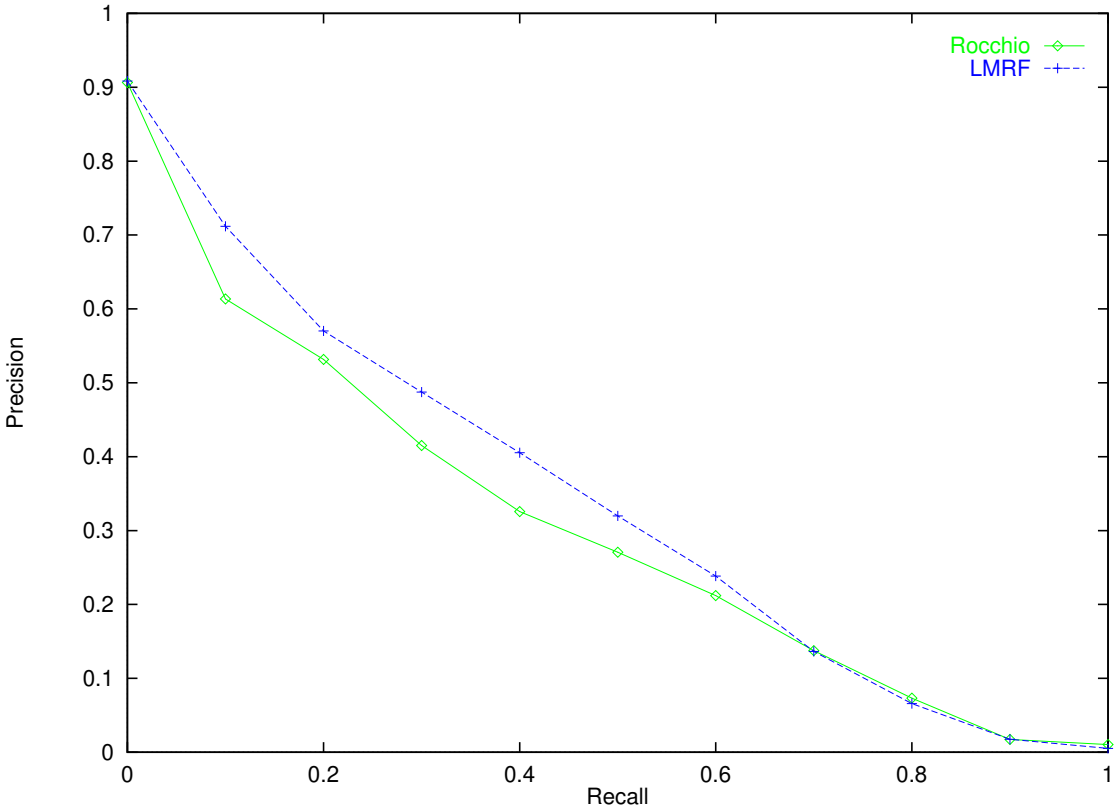

Figure 5.3. Comparison of Rocchio to the Language modeling approach using 10 documents and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

$$
\begin{aligned}
L_t &= \log\left(\frac{P_{avg}(t|d \in R)}{P_{avg}(t|d \in \overline{R})}\right) \\
P_{avg}(t|d \in R) &= \frac{\sum_{d \in R} P(t|M_d)}{|R|} \\
P_{avg}(t|d \in \overline{R}) &= \frac{\sum_{d \in \overline{R}} P(t|M_d)}{|\overline{R}|}
\end{aligned}
$$

Recall that in the relevance feedback case, a small number number of documents were known to be relevant but the relevance of the rest of the collection was unknown. In the routing case, complete relevance judgments are available for the training collection. Ratio method 2 makes use of this additional information.

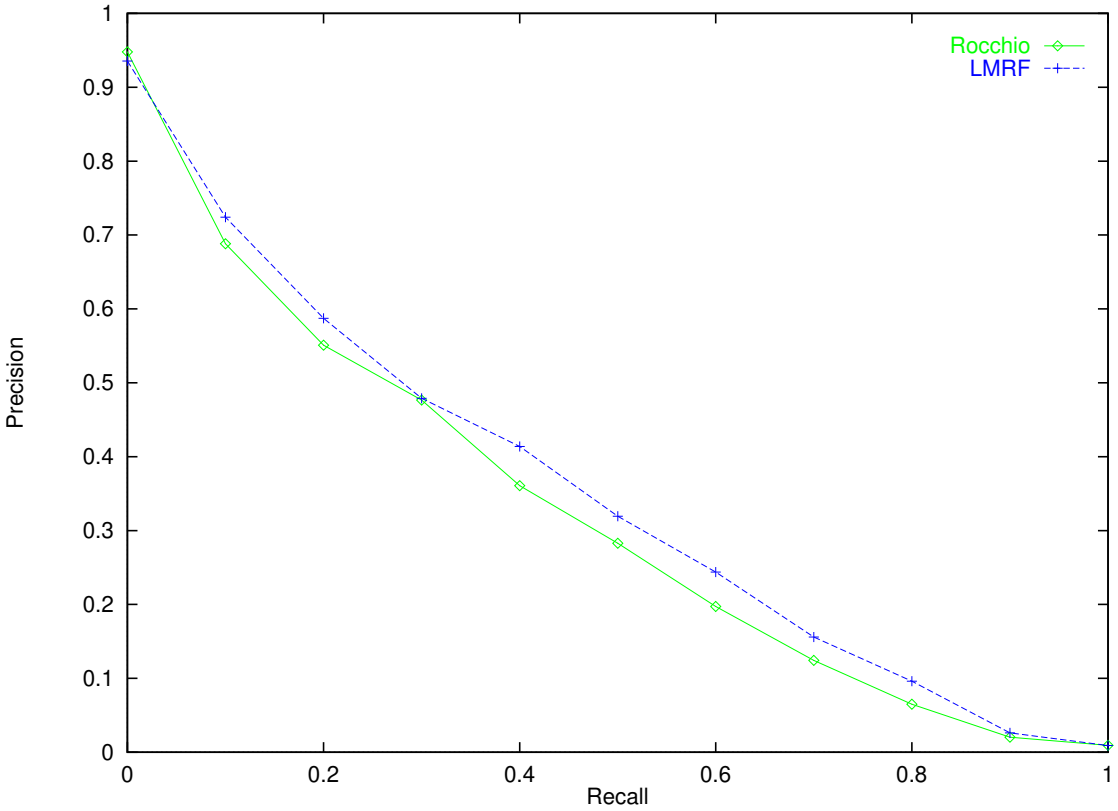

Figure 5.4. Comparison of Rocchio to the Language modeling approach using 10 documents and adding 10 terms on TREC queries 202-250 on TREC disks 2 and 3.

5.2.2.2 Routing Results

Experiments were performed on the TREC 2 routing task. The training data was TREC disks 1 and 2 and the test set was TREC disk 3. The queries were derived from TREC topics 51-100. The initial queries consisted of the terms from the concept fields. These queries were augmented by adding the top 20 terms as ranked by each of the ratio methods. The results for these two augmented query sets are shown in Figure 5.5. Notice that ratio 2 outperforms ratio 1, which is to be expected, since ratio 2 makes use of the non-relevance judgments in addition to the relevance judgments. For purposes of comparison, the official UMass TREC 2 results are also included in the graph. Notice that ratio method one performs better than the UMass run at the high precision end of the curve, is approximately equal in the middle and drops below at the high recall end of the curve. Ratio method 2 is better for most of the curve

Table 5.3. Comparison of Rocchio to the Language modeling approach using 10 documents and adding 5 terms on TREC queries 202-250 on TREC disks 2 and 3.

	Rocchio	LMRF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3834	4124	+7.56	28/44	0.0481⋆	0.0168⋆
Precision						
at 0.00	0.9064	0.9083	+0.2	9/18	0.5927	0.6763
at 0.10	0.6134	0.7117	+16.0	32/42	0.0005⋆	0.0005⋆
at 0.20	0.5317	0.5701	+7.2	28/45	0.0676	0.0431⋆
at 0.30	0.4151	0.4875	+17.4	27/46	0.1510	0.0200⋆
at 0.40	0.3257	0.4054	+24.5	29/46	0.0519	0.0037⋆
at 0.50	0.2706	0.3198	+18.2	24/39	0.0998	0.0194⋆
at 0.60	0.2120	0.2384	+12.5	26/38	0.0168⋆	0.0355⋆
at 0.70	0.1374	0.1364	−0.7	19/32	0.8923	0.7728
at 0.80	0.0731	0.0658	−10.0	12/20	0.8684	0.8047
at 0.90	0.0173	0.0175	+1.2	2/8	0.9648	undef
at 1.00	0.0103	0.0051	−50.0	0/3	0.1250	undef
Avg:	0.2943	0.3279	+11.44	31/49	0.0427⋆	0.0186⋆
Precision at:						
5 docs:	0.7061	0.7347	+4.0	14/28	0.5747	0.2545
10 docs:	0.6306	0.6878	+9.1	23/36	0.0662	0.0176⋆
15 docs:	0.5728	0.6218	+8.6	25/33	0.0023⋆	0.0160⋆
20 docs:	0.5337	0.5796	+8.6	29/40	0.0032⋆	0.0110⋆
30 docs:	0.4810	0.5156	+7.2	26/39	0.0266⋆	0.0230⋆
100 docs:	0.3169	0.3408	+7.5	26/41	0.0586	0.0284⋆
200 docs:	0.2251	0.2457	+9.2	25/42	0.1400	0.0576
500 docs:	0.1290	0.1367	+6.0	27/46	0.1510	0.0452⋆
1000 docs:	0.0782	0.0842	+7.6	28/44	0.0481⋆	0.0168⋆
R-Precision:	0.3474	0.3773	+8.61	29/43	0.0158⋆	0.0181⋆

dropping below the Umass run at the high recall end of the curve. Also see Table 5.5. For purposes of comparison, the average precision of ratio method 1 approximately ties the UMass results in TREC 2, one of the top performing systems that year.

Also notice that the average precision for ratio 2 is competitive with the best systems in TREC to date. For purposes of comparison, the best routing average precision result in any of the TREC evaluations is approximately 41% [26]. This best ever result was obtained using a search based optimization of term weights. The results presented here did not require any term weighting at all, nor were other sources of evidence, such as proximity information, often used by systems in TREC. These techniques may improve results further, but this question will be left for future work.

Table 5.4. Comparison of Rocchio to the Language modeling approach using 10 documents and adding 10 terms on TREC queries 202-250 on TREC disks 2 and 3.

	Rocchio	LMRF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3933	4210	+7.04	35/45	0.0001⋆	0.0013⋆
Precision						
at 0.00	0.9478	0.9354	−1.3	6/10	0.8281	0.4392
at 0.10	0.6882	0.7242	+5.2	24/42	0.2204	0.1482
at 0.20	0.5509	0.5873	+6.6	27/45	0.1163	0.1169
at 0.30	0.4766	0.4789	+0.5	25/46	0.3294	0.3819
at 0.40	0.3608	0.4138	+14.7	30/46	0.0270⋆	0.0037⋆
at 0.50	0.2829	0.3193	+12.9	30/45	0.0178⋆	0.0153⋆
at 0.60	0.1973	0.2439	+23.6	28/39	0.0047⋆	0.0164⋆
at 0.70	0.1242	0.1559	+25.5	22/35	0.0877	0.0442⋆
at 0.80	0.0650	0.0961	+47.9	13/21	0.1917	0.0315⋆
at 0.90	0.0203	0.0263	+29.4	5/9	0.5000	undef
at 1.00	0.0094	0.0090	−4.6	0/2	0.2500	undef
Avg:	0.3138	0.3393	+8.11	33/49	0.0106⋆	0.0190⋆
Precision at:						
5 docs:	0.7469	0.7837	+4.9	13/23	0.3388	0.2421
10 docs:	0.6653	0.6878	+3.4	20/37	0.3714	0.1709
15 docs:	0.6150	0.6354	+3.3	22/37	0.1620	0.1066
20 docs:	0.5755	0.5939	+3.2	27/46	0.1510	0.0987
30 docs:	0.5150	0.5306	+3.0	23/40	0.2148	0.1519
100 docs:	0.3300	0.3396	+2.9	31/44	0.0048⋆	0.0405⋆
200 docs:	0.2356	0.2415	+2.5	26/43	0.1110	0.1102
500 docs:	0.1327	0.1391	+4.9	29/45	0.0362⋆	0.0346⋆
1000 docs:	0.0803	0.0859	+7.0	35/45	0.0001⋆	0.0013⋆
R-Precision:	0.3550	0.3813	+7.43	30/45	0.0178⋆	0.0431⋆

These results provide more evidence that the language modeling approach is a good model for retrieval since a technique that follows from this approach yields routing performance competitive with the best results in the field.

5.3 Query Expansion Without Relevance Information

Query expansion techniques that do not make use of relevance information are used to automatically add terms to a query in order to improve retrieval performance. In the standard probabilistic models of IR, unsupervised query expansion techniques are often justified by the association hypothesis. Essentially, if two terms s and t

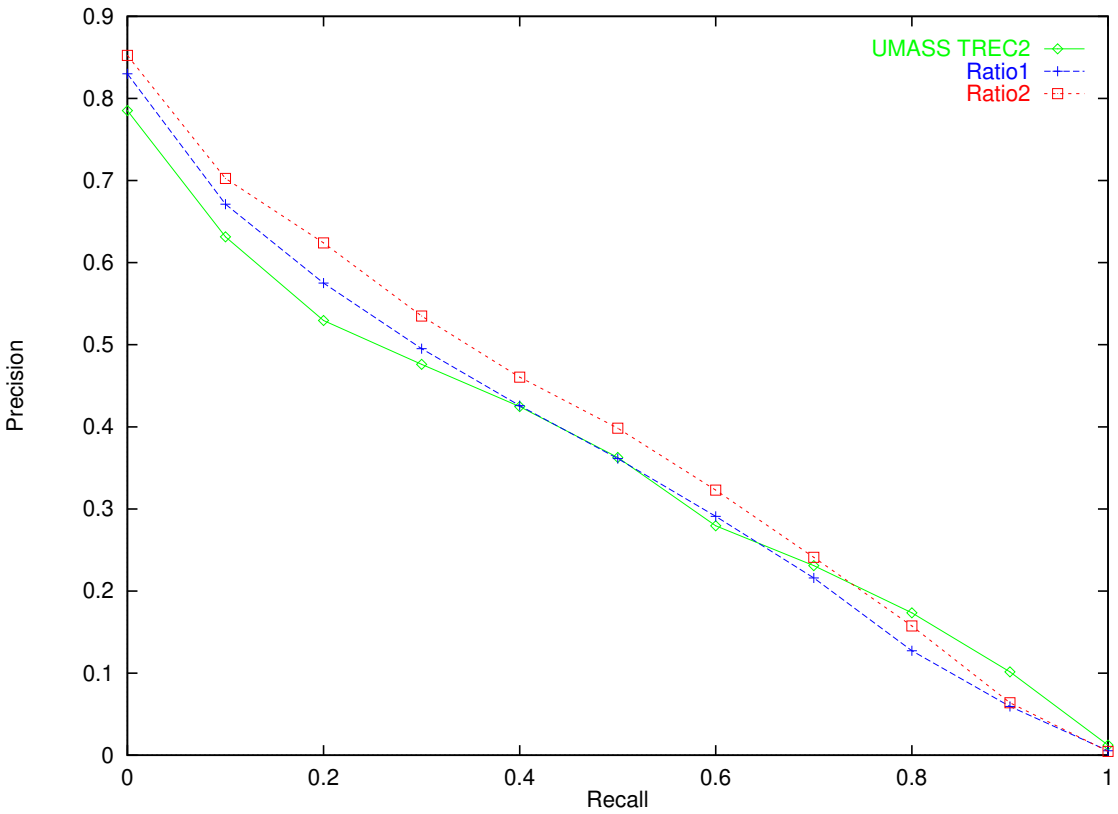

Figure 5.5. Comparison of ratio methods one and two on TREC 93 routing task.

co-occur frequently, one can infer that documents about the concept represented by term s are likely to be about the concept represented by term t as well [67].

In the language modeling approach, inferences are not made as to what documents are about and so the standard association hypothesis does not apply. However, whatever the documents are about, the language models being estimated can be used to generate additional terms to augment the initial query. These terms should be highly probable according to models of interest relative to their occurrence probability in the collection as a whole if they are chosen from the top retrieved documents.

Query expansion techniques fall into two broad classes, namely global methods and local methods. Global methods include association thesauri such as PhraseFinder [31] and Latent Semantic Indexing [50]. Local methods include local feedback and Local Context Analysis (LCA). The distinction between these two classes is that

Table 5.5. Comparison of ratio methods one and two on TREC 93 routing task.

	One	Two	% chng.	I/D	Sign	Wilcoxon
Relevant:	10485	10485				
Rel. ret.:	7246	7599	+4.87	28/39	0.0047⋆	0.0039⋆
Precision						
at 0.00	0.8300	0.8525	+2.7	8/14	0.3953	0.1653
at 0.10	0.6712	0.7026	+4.7	28/42	0.0218⋆	0.0047⋆
at 0.20	0.5752	0.6241	+8.5	32/48	0.0147⋆	0.0020⋆
at 0.30	0.4953	0.5350	+8.0	30/48	0.0557	0.0060⋆
at 0.40	0.4259	0.4605	+8.1	27/50	0.3359	0.0535
at 0.50	0.3613	0.3983	+10.2	29/48	0.0967	0.0424⋆
at 0.60	0.2909	0.3229	+11.0	28/44	0.0481⋆	0.0499⋆
at 0.70	0.2159	0.2411	+11.7	23/37	0.0939	0.0228⋆
at 0.80	0.1273	0.1575	+23.7	18/28	0.0925	0.0065⋆
at 0.90	0.0594	0.0639	+7.7	9/16	0.4018	0.2190
at 1.00	0.0055	0.0047	−14.9	1/3	0.5000	undef
Avg:	0.3543	0.3839	+8.36	34/50	0.0077⋆	0.0007⋆
Precision at:						
5 docs:	0.6880	0.7040	+2.3	12/18	0.1189	0.2040
10 docs:	0.6680	0.7020	+5.1	18/29	0.1325	0.0536
15 docs:	0.6440	0.6840	+6.2	22/32	0.0251⋆	0.0158⋆
20 docs:	0.6360	0.6590	+3.6	21/36	0.2025	0.0720
30 docs:	0.5987	0.6260	+4.6	22/34	0.0607	0.0246⋆
100 docs:	0.4612	0.4888	+6.0	29/43	0.0158⋆	0.0036⋆
200 docs:	0.3660	0.3865	+5.6	28/41	0.0138⋆	0.0063⋆
500 docs:	0.2337	0.2452	+4.9	25/41	0.1055	0.0150⋆
1000 docs:	0.1449	0.1520	+4.9	28/39	0.0047⋆	0.0039⋆
R-Precision:	0.3901	0.4171	+6.92	29/43	0.0158⋆	0.0071⋆

global methods analyze the entire collection, while local methods concentrate on the top retrieved documents for an individual query. It is local information that will be focused on here. Specifically, the techniques of local feedback and Local Context Analysis.

5.3.1 Local Context Analysis (LCA)

It was shown by Xu [67] that global methods, such as association thesauri, are effective precisely because they find concepts from the top ranked documents. In other words, local information is more useful than global information for query expansion. Also, it was shown by Xu and Croft [68] that LCA is more robust than local feedback with regards to the number of top documents to feed back.

Briefly, LCA can be described as follows. The initial query is run against a passage database and the top N passages are retrieved. The value of N must be determined empirically but, in experiments, LCA appears to be very effective over a very wide range of values (anywhere from 30 to 100 passages works well) [67].

Given these top N passages, concepts (terms and phrases) will be chosen based on their co-occurrence with the query terms in the top N passages. The formula used to rank the concepts is the following [67]:

$$bel(Q,c) = \prod_{t_i \in Q} (\delta + \log(af(c,t_i))\ idf_c / \log(n))^{idf_i}$$

with

$$\begin{aligned} af(c,t_i) &= \textstyle\sum_{j=1}^{j=n} ft_{ij}\ fc_j \\ idf_i &= min(1.0, log10(N/N_i)/5.0) \\ idf_c &= min(1.0, log10(N/N_c)/5.0) \end{aligned}$$

c	a concept
ft_{ij}	occurrences of t_i in p_j
fc_j	occurrences of c in p_j
N	passages in the collection
N_i	passages containing t_i
N_c	passages containing c
δ	0.1 – small constant to prevent zero

This formula will rank concepts that co-occur with more query terms higher than those that occur with fewer, and will give higher weight to rarer concepts co-occurring with rarer query terms than for more common concepts and query terms. This formula

is clearly heuristic in nature, but it works very well. In Section 5.3.3, a probabilistic technique similar to LCA will be developed. Note that LCA as developed by Xu and Croft uses phrase and passage information [68]. The version developed here is somewhat simplified and uses only document and term information. The starting point of this method will be probabilistic local feedback which is outlined in the next section (5.3.2).

5.3.2 Local Feedback

As mentioned in Section 2.2.8, the query expansion technique of local feedback [5] is performed by assuming that the top n documents in the initial ranked list are relevant and performing relevance feedback as usual. In Section 5.2 the ratio method of relevance feedback was introduced where terms in the relevant set of documents are ranked according to:

$$L_t = \sum_{d \in T_n} \log \left(\frac{P(t|M_d)}{\frac{cf_t}{cs}} \right)$$

where T_n is the set of the top n retrieved documents, $P(t|M_d)$ is the probability of term t given the document model as defined in Chapter 4, cf_t is the raw count of term t in the collection and cs is the raw collection size. This formula will also be used to choose terms for local feedback.

5.3.2.1 Local Feedback Results

The first experiment ranks the terms according to ratio method 1 (described in Section 5.2). Table 5.6 shows the results of feeding back the top ten documents and adding the top five terms to the initial query. Results improve at most levels of recall and average precision is improved by 23% over the baseline result. The same result is shown graphically in Figure 5.6.

Notice the slight drop in performance at the top of the ranking. This often happens in local feedback because some of the original top ranked documents were not relevant, and so feeding them back introduces some bad terms to the query. A second experiment was done that attempted to correct this problem.

Table 5.6. Comparison of baseline to unweighted local feedback.

	Orig.	LF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3364	3912	+16.29	37/43	0.0000⋆	0.0000⋆
Precision						
at 0.00	0.7590	0.7396	−2.6	14/26	0.7214	0.5051
at 0.10	0.4910	0.5629	+14.6	32/43	0.0010⋆	0.0007⋆
at 0.20	0.4045	0.4707	+16.4	30/45	0.0178⋆	0.0033⋆
at 0.30	0.3342	0.4047	+21.1	33/44	0.0006⋆	0.0004⋆
at 0.40	0.2572	0.3422	+33.0	30/42	0.0040⋆	0.0000⋆
at 0.50	0.2061	0.2715	+31.7	29/39	0.0017⋆	0.0003⋆
at 0.60	0.1405	0.2056	+46.3	24/33	0.0068⋆	0.0003⋆
at 0.70	0.0760	0.1347	+77.2	24/28	0.0001⋆	0.0000⋆
at 0.80	0.0432	0.0757	+75.2	15/19	0.0096⋆	0.0010⋆
at 0.90	0.0063	0.0112	+79.1	5/6	0.1094	undef
at 1.00	0.0050	0.0049	−1.9	2/3	0.8750	undef
Avg:	0.2233	0.2754	+23.33	36/49	0.0007⋆	0.0000⋆
Precision at:						
5 docs:	0.5020	0.5714	+13.8	19/28	0.0436⋆	0.0153⋆
10 docs:	0.4898	0.5327	+8.7	18/30	0.1808	0.0510
15 docs:	0.4435	0.4993	+12.6	24/37	0.0494⋆	0.0058⋆
20 docs:	0.4051	0.4704	+16.1	26/36	0.0057⋆	0.0004⋆
30 docs:	0.3707	0.4252	+14.7	26/38	0.0168⋆	0.0010⋆
100 docs:	0.2500	0.2994	+19.8	32/43	0.0010⋆	0.0001⋆
200 docs:	0.1903	0.2201	+15.7	30/42	0.0040⋆	0.0004⋆
500 docs:	0.1119	0.1284	+14.8	33/41	0.0001⋆	0.0000⋆
1000 docs:	0.0687	0.0798	+16.3	37/43	0.0000⋆	0.0000⋆
R-Precision:	0.2876	0.3280	+14.03	32/43	0.0010⋆	0.0008⋆

The second experiment uses the model probability to weight the contribution of each ratio to the sum. Table 5.7 shows the results of feeding back the top ten documents and adding the top five terms to the initial query. Also see Figure 5.7. Again, results improve at most levels of recall. The top point is slightly better but, the average precision has only improved by approximately 9 percent over baseline, so the additional weighting has caused the results to improve much less than the unweighted local feedback on average. This means that while this method helps to

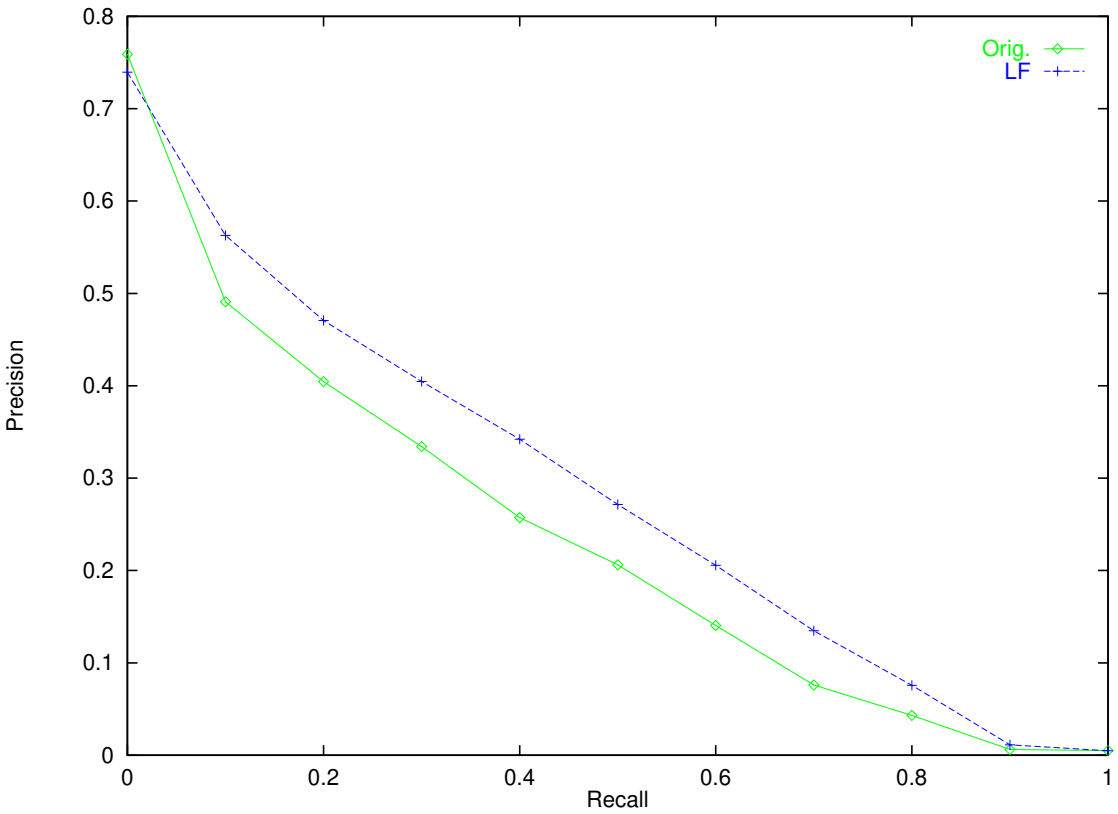

Figure 5.6. Comparison of baseline to unweighted local feedback.

correct the problem at the top of the ranking, it hurts performance overall. The next experiment will augment local feedback in a different way that will correct the problem at the top of the ranking, while maintaining good performance on average.

5.3.3 Extending Local Feedback with Co-Occurrence Information

The technique of local context analysis is arguably the most successful method for unsupervised query expansion available. A complete language modeling analog to LCA is beyond the scope of this work, but the technique described here takes a step in that direction. One of the important differences between local feedback and LCA is that LCA uses additional information about the co-occurrence of concepts with the query terms. The starting point of the technique developed here is the ratio method of local feedback described in Section 5.3.2, where each term is weighted by L_t, the

Table 5.7. Comparison of baseline to weighted local feedback.

	Orig.	LF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3364	3608	+7.25	31/44	0.0048⋆	0.0002⋆
Precision						
at 0.00	0.7590	0.7567	−0.3	11/19	0.8204	0.4520
at 0.10	0.4910	0.5244	+6.8	28/42	0.0218⋆	0.0092⋆
at 0.20	0.4045	0.4065	+0.5	22/44	0.5598	0.2456
at 0.30	0.3342	0.3485	+4.3	27/42	0.0442⋆	0.0439⋆
at 0.40	0.2572	0.2918	+13.4	24/39	0.0998	0.0088⋆
at 0.50	0.2061	0.2356	+14.3	23/34	0.0288⋆	0.0103⋆
at 0.60	0.1405	0.1807	+28.6	22/28	0.0019⋆	0.0009⋆
at 0.70	0.0760	0.1037	+36.4	21/26	0.0012⋆	0.0027⋆
at 0.80	0.0432	0.0618	+43.0	12/16	0.0384⋆	0.0170⋆
at 0.90	0.0063	0.0126	+100.8	4/5	0.1875	undef
at 1.00	0.0050	0.0058	+16.8	2/3	0.5000	undef
Avg:	0.2233	0.2424	+8.55	33/49	0.0106⋆	0.0035⋆
Precision at:						
5 docs:	0.5020	0.5551	+10.6	16/22	0.0262⋆	0.0310⋆
10 docs:	0.4898	0.5000	+2.1	15/29	0.5000	0.2411
15 docs:	0.4435	0.4558	+2.8	19/32	0.1885	0.1798
20 docs:	0.4051	0.4296	+6.0	20/32	0.1077	0.0444⋆
30 docs:	0.3707	0.3939	+6.2	21/34	0.1147	0.0421⋆
100 docs:	0.2500	0.2714	+8.6	29/43	0.0158⋆	0.0107⋆
200 docs:	0.1903	0.2024	+6.4	28/43	0.0330⋆	0.0271⋆
500 docs:	0.1119	0.1179	+5.4	27/40	0.0192⋆	0.0032⋆
1000 docs:	0.0687	0.0736	+7.3	31/44	0.0048⋆	0.0002⋆
R-Precision:	0.2876	0.2967	+3.14	25/44	0.2257	0.0376⋆

sum of the log ratio of the document probabilities in the top n documents vs. the collection probabilities.

In order to incorporate co-occurrence information, a co-occurrence language model will be estimated for the top retrieved documents. The event space of term co-occurrences will be defined as the set of events consisting of token pairs occuring in the top n documents. For example, if a query term q occurs 5 times in a document and a candidate expansion term c occurs 5 times this will count as 25 co-occurrences. This is similar to Xu's co-occurrence count [67]. These counts will be summed for each (q, t) pair, i.e., for each query term paired with every other term in the top retrieved documents, and for each query term q. The totals $\#(q, t)$ and $\#q$ will be

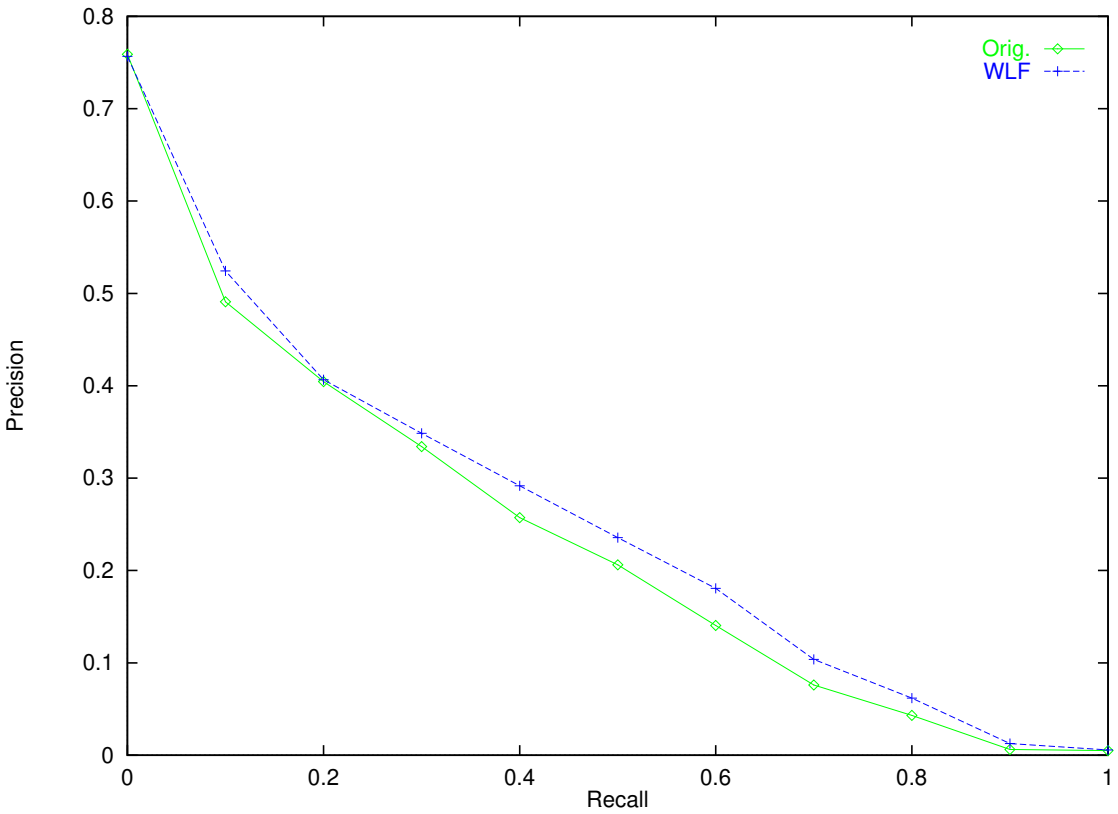

Figure 5.7. Comparison of baseline to weighted local feedback.

used to estimate $\hat{p}(t|q)$, the probability of t occurring given an occurrence of q, as follows:

$$\hat{p}(t|q) = \frac{\#(q,t)}{\#q}$$

This probability is used to weight the candidate terms as follows:

$$w_t = \prod_q L_t \times \hat{p}(t|q)$$

This product essentially adds a constraint to the local feedback technique used earlier. Each term is weighted according to the co-occurrence probability. Again, it should be pointed out that LCA has additional components not present in this technique. These components include passage level, rather than document level, co-

occurrence statistics, phrases, and within-query weighting of the added features. It is hoped that further development of this technique will eventually lead to a complete, rigorous probabilistic description of LCA. However, this is left for future work.

5.3.3.1 Harper and van Rijsbergen's Co-Occurrence Model

A related use of co-occurrence information was that developed by Harper and van Rijsbergen, previously discussed in section 2.2.7.1. This work was an attempt to relax the assumption of the independence of query terms in the estimation of the joint distribution of query terms in relevant documents. The event space was the occurrence of a term one or more times in a given document, i.e., within document term frequency was not modeled. In order to approximate the dependence relationships in a tractable manner, a tree was built with nodes representing each of the query terms. The edges of the tree were weighted by the expected mutual information and the tree was chosen that spanned all of the nodes and maximized the total expected mutual information [27].

The extended local feedback technique described here differs from Harper and van Rijsbergen's work in ways. First, while Harper and van Rijsbergen used the probability estimates to re-rank documents in the context of relevance feedback, the task in extended local feedback was to choose new terms for query expansion without relevance information. Second, the co-occurrence model used here accounts for the number of within document co-occurrences, rather than just the presence or absence of one or more co-occurrences in a document.

5.3.3.2 Extended Local Feedback Results

To test the new technique, TREC queries 202-250 were run against TREC disks 2 and 3 as before. The top ten documents were fed back and the top ten terms were added to the query and the expanded query was rerun. The results are shown in graphical form in Figure 5.8. and in tabular form in Table 5.8.

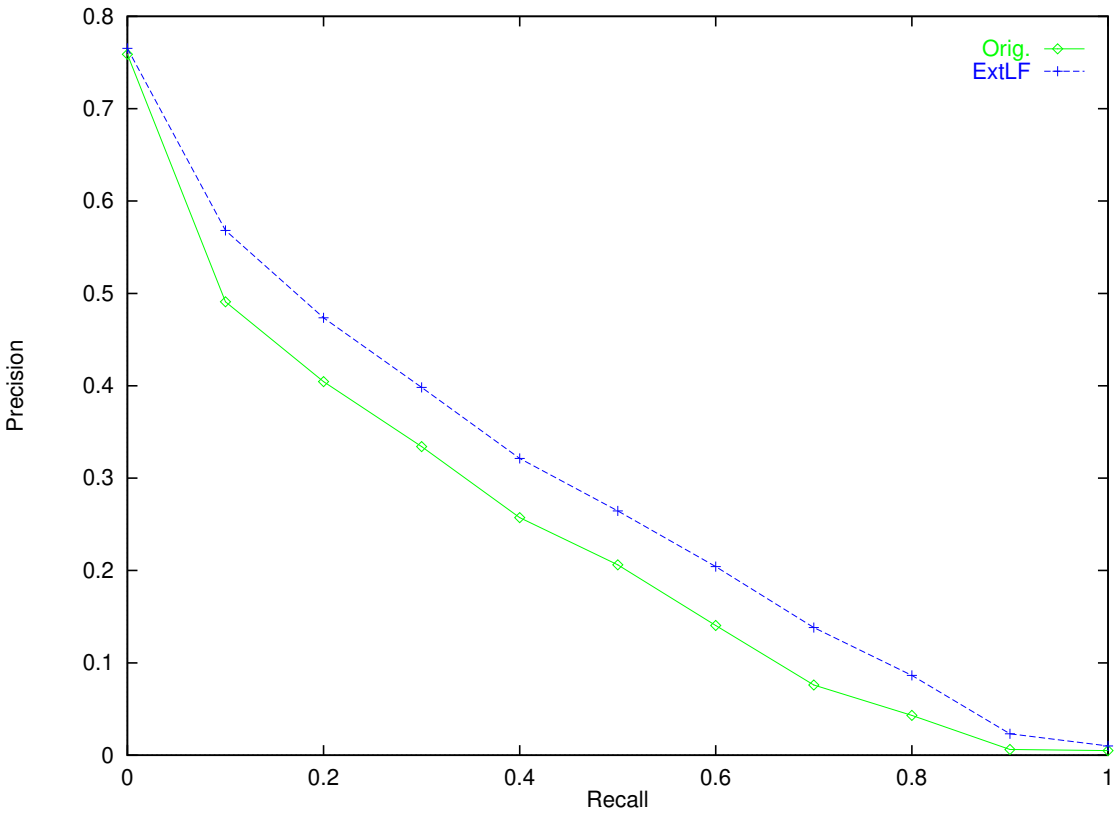

Figure 5.8. Comparison of baseline to extended local feedback.

Notice the improvement in recall, average precision and precision at all levels of recall. Also notice that in comparison to the original local feedback run shown in Table 5.6 the drop in precision at the top point has been eliminated and the results have improved on average. The 24% improvement in average precision is lower than the improvements reported by Xu for LCA, but, as mentioned, the current technique does not use within-query term weighting, passage level evidence, or phrases. It is likely that this information would improve the results, but this matter will be left for future work.

5.3.4 Non-query Terms as Evidence

An alternative to query expansion in the language modeling approach is to use co-occurring terms to obtain better estimates of the probability of each term. This would require a language model to estimate $p(q|D)$ where q is a query term and D

Table 5.8. Comparison of baseline to extended local feedback on TREC queries 202-250 on TREC disks2 and 3.

	Orig.	ExtLF	% chng.	I/D	Sign	Wilcoxon
Relevant:	6501	6501				
Rel. ret.:	3364	3711	+10.32	30/42	0.0040⋆	0.0002⋆
Precision						
at 0.00	0.7590	0.7654	+0.8	12/24	0.5806	0.3187
at 0.10	0.4910	0.5682	+15.7	29/42	0.0098⋆	0.0012⋆
at 0.20	0.4045	0.4737	+17.1	29/42	0.0098⋆	0.0033⋆
at 0.30	0.3342	0.3983	+19.2	30/42	0.0040⋆	0.0003⋆
at 0.40	0.2572	0.3213	+24.9	27/39	0.0119⋆	0.0007⋆
at 0.50	0.2061	0.2645	+28.4	27/36	0.0020⋆	0.0005⋆
at 0.60	0.1405	0.2042	+45.4	27/33	0.0002⋆	0.0001⋆
at 0.70	0.0760	0.1383	+81.9	19/24	0.0033⋆	0.0005⋆
at 0.80	0.0432	0.0864	+99.7	13/17	0.0245⋆	0.0011⋆
at 0.90	0.0063	0.0232	+271.2	8/8	0.0039⋆	undef
at 1.00	0.0050	0.0100	+102.8	2/3	0.5000	undef
Avg:	0.2233	0.2772	+24.12	35/49	0.0019⋆	0.0000⋆
Precision at:						
5 docs:	0.5020	0.6082	+21.1	20/26	0.0047⋆	0.0021⋆
10 docs:	0.4898	0.5490	+12.1	24/33	0.0068⋆	0.0068⋆
15 docs:	0.4435	0.5143	+16.0	26/37	0.0100⋆	0.0015⋆
20 docs:	0.4051	0.4745	+17.1	29/42	0.0098⋆	0.0006⋆
30 docs:	0.3707	0.4333	+16.9	28/45	0.0676	0.0021⋆
100 docs:	0.2500	0.2922	+16.9	30/46	0.0270⋆	0.0015⋆
200 docs:	0.1903	0.2144	+12.7	30/46	0.0270⋆	0.0016⋆
500 docs:	0.1119	0.1251	+11.8	29/43	0.0158⋆	0.0007⋆
1000 docs:	0.0687	0.0757	+10.3	30/42	0.0040⋆	0.0002⋆
R-Precision:	0.2876	0.3186	+10.76	29/45	0.0362⋆	0.0115⋆

is a representation of the set of observed documents. One model that could be used for this purpose is the exponential family studied by Della Pietra *et al.* [18]. Recall from Chapter 2 that this family of models takes the following form:

$$\hat{p}(q|D) = \frac{e^{\lambda \cdot f(D)} p_d(q|D)}{\sum_{\forall D} e^{\lambda \cdot f(D)} p_d(q|D)}$$

In this case, $p_d(q|D)$ is a default estimate for the probability of a term given the observation of a set of documents. Presumably this would be some function of the estimates for $\hat{p}(q|M_d)$, the probability of generating q from each document model as described in Chapter 4, perhaps the average over the set in question. The feature functions $f(D)$ would take the form "term t occurs in document d with probability

$\hat{p}(t|M_d)$." These features would be induced according to the algorithm described in [18] and the parameter vector λ would need to be estimated from the data.

Currently, the estimation problem for this distribution is too time consuming to be practical for query expansion in an interactive or routing environment. In order to develop this idea, a tractable approximation would need to be developed, or a tractable multiple predictor model would need to be developed, but these matters are beyond the scope of this thesis.

CHAPTER 6

ADDITIONAL ASPECTS OF RETRIEVAL

This chapter begins with an introduction to additional features used by retrieval systems and additional problems associated with retrieval.

Following that, the use of proximity information is discussed and then, in Section 6.2, a distinction is made between two different types of proximity information, phrasal evidence and passage level evidence. These two sources of evidence will be discussed in the context of the language modeling approach in sections 6.2.2 and 6.2.4 respectively.

Section 6.3 addresses the issue of Boolean queries and their interpretation in the context of the language modeling approach. In addition, generalizations of Boolean operators are discussed.

Finally, Section 6.4 discusses the problem of within query term weighting and the implications that the language modeling approach has for this problem.

6.1 Chapter Introduction

So far, the language modeling approach to retrieval has been developed using only single word features. This is more reasonable than one might first expect since query terms tend to disambiguate each other. For example, given the query, "base ball bat pitcher," it is very likely that the user wished to retrieve documents about baseball, even though each of the four words is ambiguous individually.

Nevertheless, retrieval systems use additional features including phrases, passage level evidence, and Boolean operators to provide more effective retrieval. Making

use of these features requires, in general, a method of assigning appropriate weights. Most retrieval system use some variant of *tf.idf* weighting. Recall that, for single term features, *tf* is a function of the within document frequency of a term and *idf* is a function of the number of documents in which a term occurs.

In the context of the language modeling approach to retrieval, the choice of feature weights will often have a natural interpretation in terms of probability models. This will be discussed for proximity features of various types and for Boolean features. In addition, the problem of query term weighting, used in most modern retrieval systems, will be discussed from a probabilistic perspective.

6.2 Proximity Information

The view taken here is that proximity information is used in two different ways in information retrieval. For the purposes of discussion, these will be referred to as phrasal evidence and passage level evidence. Both of these phrases can be somewhat misleading and so more precise definitions are given below.

Both sources of evidence being referred to are statistical in nature, i.e. phrases refer to combinations of words identified by occurrence statistics as opposed to actual linguistic phrases. Likewise, passages refer to identifiable units of text which can be either identified statistically as was done in Chapter 3, or simply fixed sized windows of text.

6.2.1 New Features vs. New Models

These sources of evidence are distinguished by the following. Phrases will refer to additional features of the documents. Their probabilities will be estimated in much the same way as term probabilities are estimated now. On the other hand, passages will be used to reduce 'false hits' in multi-topic documents by requiring query terms to appear in proximity. That is, passage level evidence reflects the uncertainty about

topical cohesion in documents or, in some sense, the uncertainty about quality of the segmentation imposed by the document boundaries.

These differences can be summed up as follows: phrasal evidence refers to the estimation of the probabilities of an expanded set of features while passage level evidence refers to the estimation of feature probabilities from an enhanced set of document models. The methods of estimation will now be defined.

6.2.2 Phrasal Evidence

Phrasal evidence should be regarded as an extended feature set. In the language modeling approach, this would mean that the probability of generating the query would be the probability of generating each phrase rather than each word (or perhaps in addition to each word).

For example, consider the phrase 'black ice' meaning a very thin coating of ice on pavement. The phrase can be thought of as a lexeme, i.e., the two words together make up a single lexical item, because its meaning cannot be derived from the meanings of the individual words. If one had a way of identifying phrases in queries (including asking the user to identify them), they may be better features than the words themselves.

In order to incorporate phrasal evidence, one needs a way of combining it with the term based evidence. There are two ways one might do this. First, phrase identification can be viewed as a segmentation problem analogous to the Chinese segmentation problem [42]. The collection can be indexed by the segmented lexical items that will treated in exactly the same way as terms. This may be acceptable in a routing environment where the query is determined by system based on prior relevance judgments. In this case, if a better feature set can be produced, the system may be able to take advantage of it. Indexing in this manner follows one of the indexing recommendations of Salton, previously described in Section 4.2. However,

indexing by phrasal features is only a good idea under the assumption that queries and documents are fundamentally the same kind of object. When users are involved, indexing by phrasal features is not a good idea, since it is likely that users will not always agree with the automatic segmenter with respect to words and phrases. Instead, an approach based on backoff models can be used. This approach would be reasonable in both routing and interactive environments.

6.2.3 Backoff Models

One may wish to include phrasal information in order to enhance precision. For example, if a user were to identify the phrase "South Africa," in a query, it is likely that the phrase would help the system return fewer false hits than the words "South" and "Africa" individually. On the other hand, consider the phrase "information retrieval." If one were to place a strict requirement on the phrase, documents mentioning "retrieval in text-based information systems" would be missed if they did not also contain the phrase "information retrieval." This problem was identified by Krovetz as the "partial credit" problem.

Krovetz recommended linguistic analysis to determine cases where credit should be assigned to individual components of a phrase [35]. Linguistic analysis may well be appropriate, but it is not entirely clear how one would assign *tf.idf* weights to the individual components, as well as to the phrase as whole, to maximize retrieval effectiveness.

This is a difficult problem, but the language modeling approach suggests a solution. Backoff models allow the system to use the extra evidence in the phrase without being hurt by it. The model is defined as follows for a two term phrase:

$$\hat{p}(w_1 w_2|M_d) = \beta_2 p(w_1 w_2|M_d) + \beta_1 p(w_1|M_d) + \beta_1 p(w_2|M_d) + \beta_0 p(\text{default})$$

In this formula, $p(w_1 w_2|M_d)$ is the estimated probability of the two word phrase given the document model. The probabilities $p(w_1|M_d)$ and $p(w_2|M_d)$ are the individual word probabilities given the document model and the β_i are the backoff parameters. For a non-occuring phrase, this estimate "backs off" to the individual word probabilities and, similarly, for non-occuring words, it "backs off" to a default estimate. This model is analogous to the current single word method described in chapter 4 where the collection probability is used as the default estimate.

The phrase probability will be a weighted average of the estimate of the phrase as a whole, along with each term individually. This method has the advantage that documents will score higher for containing the phrase but will still receive some credit for the individual terms. Standard methods for the estimation of backoff model parameters can be used to determine the weighting, as described in section 2.1.3. It should be pointed out that any sort of proximity feature would fall under the heading of what has been referred to here as phrasal evidence, and a backoff model could be defined similarly for any of these features.

If one wished to perform linguistic analysis, as recommended by Krovetz, one could incorporate this knowledge into the language models. For example, Krovetz refers to words that appear in many semantically unrelated phrases as *promiscuous* words, and recommends against assigning partial credit to such words when they appear in a query as part of a phrase. If one had a method of identifying words with this property (perhaps using mutual information statistics), one could build this knowledge into the backoff scheme. The interpolation weights for the promiscuous words would be lower than those for the less promiscuous words as follows:

$$\hat{p}(w_1 w_2|M_d) = \beta_2 p(w_1 w_2|M_d) + \beta_{1_p} p(w_1|M_d) + \beta_1 p(w_2|M_d) + \beta_0 p(\text{default})$$

where, w_1 is a promiscuous word and w_2 is a non-promiscuous word. The interpolation weight for w_1, β_{1_p}, would be lower than β_1 the interpolation weight for

w_2, as determined by the counts in related phrases normalized by the counts in all phrases. Whether the promiscuity of phrase components is possible to identify, or whether this property is useful for retrieval, is unknown. The point of this example is that when one has knowledge about the characteristics of language, or perhaps just an interesting idea, one can easily incorporate this knowledge, or this idea, into a language model and, by means of the language modeling approach, into the process of retrieval. This allows the idea to be evaluated in a controlled manner by comparing, in this example, the results of the promiscuity model to the model that treats all words identically.

6.2.4 Passage Level Evidence

This section discusses a range of possible probabilistic techniques for the incorporation of passage level evidence into the retrieval process. Empirical studies will be left for future work, and are discussed further in Section 7.2.6.

In order to incorporate passage level evidence, each document model can be viewed as a mixture of individual passage models. The probability of producing the query given the document model can then be a simple sum of the probabilities of producing the query given each passage, divided by the number of passages, or it can be a more complex mixture giving, for example, additional weight to the better passages. Much like the phrasal evidence problem, one can view passage identification as a segmentation problem. Using the method described in Chapter 3, one can identify cohesive passages in the text and use a language model for each. One could also use fixed sized passages which have been shown to perform well in practice [12, 1].

6.2.4.1 Sliding Window Passages

A natural extension to the fixed sized passage technique is to use shifting windows in order to prevent error due to incorrect boundaries. This can be improved further by means of a kernel style estimator where the width of the kernel is determined by

the desired passage size and the probability of individual words occurring in the same passage will be determined by the kernel function.

As mentioned, it is also possible to use a weighted sum where better passages contribute more to the score so that multi-topic documents with interesting sections will still receive reasonable scores. In order to make this method more robust, a backoff scheme can be implemented where the passage scores will be mixed with the document score.

A related problem is that of passage retrieval where the task is to retrieve only relevant passages rather than full documents. The techniques discussed here may provide the means for effective passage retrieval. In order to test them, test collections with passage level judgments would need to become available. This matter is discussed further in Section 7.2.6.

6.3 Boolean Queries

It should be noted that the language modeling approach as currently defined essentially implements a probabilistic Boolean AND over the query terms and a probabilistic AND NOT over the non-query terms, assuming independence of terms. The probabilistic AND is implemented as the product of the individual probabilities of the event and the probabilistic NOT is defined as one minus the probability of the event.

Implementation of probabilistic OR over two terms a and b takes the form:

$$(1.0 - ((1.0 - p(a)) \times (1.0 - p(b))))$$

Arbitrarily complex Boolean features could, in principle be constructed from these three operators, but it is not likely that users will have good intuitions for the likelihoods of arbitrary Boolean features in documents of interest.

In order to support Booleans queries without being hurt by poor user intuitions, one can give credit to documents that do not strictly match the Boolean query. Previously, in Section 6.2.3, a similar problem was addressed for phrasal features. In order to estimate the probability of a phrase, the use of a backoff model was discussed which estimates the phrase probability from both the phrase statistics and the statistics of the individual components.

In the current implementation, the query probability can be regarded as an AND. For non-occuring terms, the collection probability is used as a backoff estimate for each term. In general, one may wish to define backoff models for the different Boolean combinations in order to prevent overly strict Boolean queries from excluding too many documents, while still favoring those documents that conform to the strict interpretation of the operators. This will be discussed further in Section 7.2.3.

6.4 Query Term Weighting

Within-query term weighting is used by most modern retrieval systems. The language modeling approach, as currently defined, does not include term weighting. However, there are extensions that can be made to the model to incorporate query term weighting in a probabilistically justified manner. Two possibilities will be considered here.

6.4.1 Risk Functions

The current method of probability estimation uses the maximum likelihood probability and the average probability, combined with a geometric risk function. A simple implementation of term weighting is to modify the risk function. This can be accomplished by means of a Bayesian prior over the term distribution which forces the less important terms to be weighted more heavily by the average probability. This makes the documents less distinguishable from each other based on occurrences of the less

important terms. For more important terms, the maximum likelihood term will be allowed to dominate, making the more important terms better able to distinguish between documents. The implementation of this idea would change the ranking formula slightly. For a non-occurring term, the current ranking function estimates the probability as $\frac{cf_t}{cs}$, where cf_t is the raw count of term t in the collection and cs is the the total number of tokens in the collection.

The change will be to mix the estimate for non-occuring terms with the mean in the same way the maximum likelihood estimator is currently for terms that do occur. The intuitive meaning of this idea is that the current risk function treats all terms equally, but with prior knowledge (or prior belief) about the relative importance of terms, one can vary the risk of relying on the mean according to the prior belief of the term importance. For example, suppose a term is deemed to be completely useless, the risk function would be modified so that all of the weight is assigned to the mean. The result is that this term is assigned an equal probability estimate for every document, causing it to have no effect on the ranking. A stopword such as 'the' would fall into this category. One can regard the differences in observed values as pure noise in this case.

6.4.2 User Specified Language Models

Currently, queries are treated as a specific type of text produced by the user. One could also allow the user to specify a language model for the generation of query text. The term weights are equivalent to the generation probabilities of the query model. In other words, one would generate query text according to the probabilities and, conceptually, the query is the result of generating a large amount of query text which could then could be processed using the current method. This probably makes more sense in a routing environment where there is enough data from which to estimate the probabilities, but it could conceivably be used for *ad hoc* retrieval if users have

intuitions about the probabilities. See Section 7.2 for additional remarks on this point.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This chapter begins with a discussion of topic segmentation. Following that, future work regarding the retrieval model is discussed in Section 7.2. First, some comments about user preferences regarding retrieval systems is presented in section 7.2.1. Following that, a discussion of probability estimators, a key concept for the language modeling approach, is presented in Section 7.2.2. Boolean retrieval, first discussed is Section 6.3 is discussed further in Section 7.2.3.

Next, additional comments regarding feature selection and weighting in general are made in Section 7.2.4. Following that, a specific example is presented in Section 7.2.4.2. The chapter continues with a discussion of relevance feedback and routing in section 7.2.5. Finally, the chapter concludes with a discussion of passage retrieval in Section 7.2.6.

7.1 Text Segmentation

The new event detection task appears to be robust to the level of error in the segmentation process. Further work is needed to determine whether this is true in general. In any event, it may be desirable to achieve a lower error rate for user interface purposes. In addition, the data for the study consisted of text and hand-transcribed audio data. The segmentation of automatically transcribed audio data may be more difficult. In the text case, it was shown that this problem can be be solved by purely statistical means without resorting to expensive knowledge-based

approaches and it is likely that this would also be the case for automatically recognized speech.

During the study, two complementary feature sets were identified: discourse features and content features. Discourse features are any features that predict a break regardless of the actual topic being discussed. Content features measure changes in topic specific language. It was shown that these features are complementary and that in combination they are more effective than either of them used alone.

It was also shown that Local Context Analysis can be used to provide an improved set of content features that do not require repeated words in order to predict topic changes, whereas a technique such as text tiling [29] clearly does require repeated words to predict boundaries effectively.

7.2 Information Retrieval

A novel way of looking at the problem of text retrieval based on probabilistic language modeling has been developed. This model is simpler than many of the existing probabilistic models because it does not require a separate model of indexing apart from the retrieval model itself. This approach does not require heuristic methods of estimation. The performance of this model on the retrieval task indicates that it performs well and that improving the estimation techniques leads to improved retrieval performance. In this model, the study of information retrieval becomes the study of probability estimates. This allows the full power of probability theory and statistical inference to be brought to bear on the retrieval task.

It was also shown that a relevance feedback technique derived from the language modeling approach works as well as existing approaches to relevance feedback and does so without relying on *ad hoc* or heuristic approaches. Similar results were shown for the technique of local feedback. The language modeling approach will provide

effective retrieval and can be improved to the extent that the following conditions can be met:

1. The language models are accurate representations of the data.

2. Users understand this approach to retrieval.

3. Users have some sense of term distribution.

Condition one has been met reasonably well by the approach described in Chapter 4. However, it is likely that the estimation can be improved.

Regarding point two, the language modeling approach can be explained to users at an intuitive level and the understanding of it will facilitate the formation of better queries. Users are typically instructed to pose natural language descriptions of their information needs as queries. A user that understands the language modeling approach will tend to think in terms of which words will help the system distinguish the documents of interest from everything else. If users think in this manner they will be able to formulate queries that will better express their information needs in a manner useful to the retrieval system.

7.2.1 User Preference

It should be pointed out that while modern partial match retrieval systems consistently outperform Boolean systems in terms of retrieval effectiveness, some expert users do not like them. For example, reference librarians have reported that they do not understand why some documents were retrieved by partial match systems [11]. Boolean systems give users a sense of control over the retrieval process. As mentioned, users are typically instructed to compose a query describing their information need in natural language and the system will do the rest. Instructing users in the art of choosing good query terms both gives them a sense of control, and helps them provide the system with better queries.

Regarding point three, in order for users to identify useful words, they would benefit from a sense of how the words are distributed in the collection. A variety of both textual and graphical tools could be developed to help users get a better sense of the distribution of terms and, if desired, of more general features.

7.2.2 Estimators

In the future, the estimation problem should be looked into further. For example, in place of the histogram estimator, perhaps a kernel estimator [41] would be better. Also the use of a geometric risk function, while a reasonable choice, could perhaps be improved upon. A data transformation technique may be appropriate since the maximum likelihood estimates are not normally distributed.

Finally the estimate of default probability should be addressed. As mentioned, the current estimator could in some strange cases assign a higher probability to a non-occurring query term. This could only happen in cases of very commonly occuring terms, i.e., terms which are not likely to be useful, however, this problem should be addressed in order to insure the robustness of the estimator in such cases. The approach will be to apply a smoothing based estimator [6] that will guarantee a default probability estimate that cannot exceed the lowest estimate assigned to a document in which a given term occurs.

7.2.3 Generalized Boolean Operators

Section 6.3 discussed Boolean queries and introduced the idea of backoff models in the context of Boolean queries to prevent overly strict Boolean queries from hurting performance.

In Section 2.2.3, two classes of generalized Boolean operators were discussed. The first was the P-Norm model, developed to support generalized Boolean retrieval for the vector space model by Salton *et al.* [53]. The second was a family of similar

operators for the support of generalized Boolean retrieval for the INQUERY model, known as the PIC operators, developed by Greiff *et al.* [24].

The P-Norm model is not a probabilistic model. The reason for its effectiveness is that it adds an additional degree of freedom to the ranking formula. With an appropriately set parameter, one can get better results unless the current ranking formula is exactly optimal for the data in question. However, there is no principled way to set the parameter or any justification as to why a parameter value determined empirically on one data set should work on a new data set. On the other hand, the PIC operators combine the 'aboutness' probabilities of features in a probabilistically justifiable way. However, as with the P-Norm model, the parameters have to be set empirically. Recall/precision experiments are done and the parameters are chosen according to the results of those experiments.

Viewing generalized Boolean operators from the perspective of backoff modeling, one could estimate the interpolation parameters from a set of Boolean queries with relevance judgments. The statistics of the actual term occurrences vs. the strict Boolean combination specified by the user can be used to determine an appropriate weighting for partially satisfied Boolean queries. The resulting model would have the intuitive meaning of accounting for the difference between the user's intuition about the presence of Boolean features in documents of interest vs. the actual occurrence. Collecting a sufficient population of Boolean queries with relevance judgments would be necessary to estimate a reasonable model.

7.2.4 Feature Selection and Weighting

Over the long term, the language modeling approach may prove useful for the evaluation of feature selection and weighting techniques. For example, consider the problem of Chinese text retrieval. The Chinese language is written without inter-word delimiters such as whitespace. Most Chinese information retrieval systems make use

of a segmenter to predict the word boundaries. However one could also use a character n-gram approach. The question is whether the added complexity of segmentation will yield improved retrieval performance.

7.2.4.1 Recall/Precision Experiments

In order to answer that question, one could run recall/precision experiments comparing the segmenter to the baseline character n-gram approach. In running this experiment, if the segmenter does not outperform the n-gram approach, the researcher might conclude that segmentation is not useful. However, that is not the only possible explanation. Perhaps a different weighting scheme is needed to take advantage the segmenter. The result of the recall/precision experiment does not inform the researcher which of those two alternatives is correct, nor does it suggest how to determine the optimal weighting scheme for the segmentation based approach. However, using the language modeling approach, one could determine the usefulness of the word segmentation algorithm and the optimal weighting as follows.

7.2.4.2 Mixture Models

Assuming a probabilistic segmenter, the segmenter distribution can be used as a language model. This model can be combined with a probabilistic n-gram model (the baseline technique in the recall/precision experiment mentioned earlier) using, for example, linear interpolation. The interpolation parameters can be adjusted using recall/precision experiments to determine if the segmenter is useful and, if so, how to weight the features. Of course, linear interpolation may be the wrong tool for the job. Exploratory analysis will be required to determine reasonable methods of evidence combination for this class of experiments. When that has been determined, if segmentation is useful, there will be some range of parameters for which retrieval performance improves. Moreover, that range will determine the optimal feature weighting thereby answering both questions.

Using a similar approach with existing retrieval models is more difficult. When one uses a mixture model in this manner, the feature set is an aggregate of character n-grams of different lengths. Each character of the text can contribute to the score of a document in more than one way, depending on the model. It is no longer clear how one calculates the 'term' frequency or document frequency of these more complex features but with the language modeling approach, the models themselves determine the weighting for each document without further effort.

7.2.4.3 Stemming

A similar analysis could be done for any feature selection technique that can be described by means of a probability model and most natural language processing techniques can be described in this manner. For example, stemming is typically done by conflating variant word forms to a single stem. This is certainly not the only way to do stemming. Word morphology is an uncertain process due to sense ambiguity among other considerations.

One question that has arisen in the work of Krovetz [35] is the optimal level of 'aggressiveness' of a stemmer i.e., given a word with multiple suffixes (or perhaps prefixes), one can choose how many to remove. By removing suffixes aggressively, the effect is to conflate more words to the same stem. Any choice one makes in this regard is arbitrary. A more principled approach is to stem at all levels of aggressiveness and to learn a backoff model of the stem and suffix space. This model can then be combined with a character n-gram backoff model in much the same way as previously described for the Chinese experiment and the results will indicate the usefulness of the technique of stemming, an open question in the field of information retrieval to date, as well as the appropriate feature weighting.

7.2.5 Relevance Feedback and Routing

The experiments already performed show that the language modeling approach to relevance feedback and routing is a reasonable alternative to existing methods. In the future, additional empirical study will need to be done to determine practical considerations such as the number of terms to add to a query. In addition, the term weighting methods discussed in section 6.4 will be incorporated into the relevance feedback method and additional empirical studies will be performed to determine the utility of these techniques in practice.

7.2.6 Passage Level Evidence and Passage Retrieval

Previous experiments with passage retrieval showed that considering the best passage in a document as an additional source of evidence improves retrieval effectiveness [12, 1]. However, since the incorporation of Robertson's *tf*, passage level evidence has not yielded much of an improvement over baseline in experiments with the INQUERY system [2]. It is not clear why a new *tf* score would make passage level evidence less useful. In order to answer this question, the techniques of passage retrieval sketched in Section 6.2.4 will need to be fully developed.

Recall that in Section 6.2, passage level evidence was presented as a method of modeling the uncertainty about topical cohesion in documents. Part of the difficulty of determining the effectiveness of passage level evidence is that this uncertainty cannot, in general, be measured. The approach taken in the TDT pilot study was to assume that the original document boundaries were the ground truth for evaluation of the segmenters, but, of course, it is the degree to which the boundaries are *not* ground truth that would need to be modeled.

7.2.7 Simulating Passage Retrieval

Sanderson has developed an elegant solution to a different problem of uncertainty modeling. The problem investigated was the effects of word sense ambiguity on re-

trieval. Sanderson investigated this question not by resolving word sense ambiguity, a difficult problem, but by introducing additional ambiguity by means of what he called 'pseudo-words.' A pseudo-word is a conflation of n random words. By performing these conflations one can add a measurable amount of additional ambiguity to a collection of documents to study the effects of sense disambiguation in a controlled manner [56].

One could use a similar approach to add the 'boundary ambiguity' described earlier. For example, starting with a set of documents judged relevant to a query, remove the boundaries on either side of each document thereby adding a known amount of unrelated text. One can then study the effects of passage retrieval in this controlled setting to fully develop the methods described in section 6.2.4 and to determine their relative effectiveness. Once conclusions have been drawn about passage level evidence in the controlled setting, the question can be addressed empirically in the general setting.

The same technique can be used for the evaluation of passage retrieval. The results would only be approximate since the original documents might themselves not be entirely cohesive, but such uncertainty can be modeled using an error metric such as the one discussed in Section 3.2.3.1. These matters will be left for future work. It is hoped that data with relevance judgments at the passage level will become available in the future so that simulations will not be necessary.

BIBLIOGRAPHY

[1] Allan, James. Relevance feedback with too much data. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1995), pp. 337–343.

[2] Allan, James. Personal communication, 1997.

[3] Allan, James, Carbonell, Jaime, Doddington, George, Yamron, Jonathan, and Yang, Yiming. Topic detection and tracking pilot study: Final report. In *Proceedings of the Broadcast News Transcription and Understanding Workshop* (1998), pp. 194–218.

[4] Allan, James, Papka, Ron, and Lavrenko, Victor. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1998).

[5] Attar, R., and Fraenkel, Aviezri S. Local feedback in full-text retrieval systems. *Journal of the ACM 24*, 3 (July 1977), 397–417.

[6] Bahl, Lalit R., Jelinek, Frederick, and Mercer, Robert L. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1983), 179–190.

[7] Barnett, James. Personal communication, 1995.

[8] Beeferman, Douglas, Berger, Adam, and Lafferty, John. Text segmentation using exponential models. In *Proceedings of Empirical Methods in Natural Language Processing* (1997).

[9] Bookstein, Abraham, and Swanson, Donald. Probabilistic models for automatic indexing. *Journal for the American Society for Information Science 25*, 5 (1976), 312–318.

[10] Buckley, Chris, Singhal, Amit, Mitra, Mandar, and Salton, Gerard. New retrieval approaches using smart: Trec 4. In *Proceedings of the TREC 4 Conference* (1995).

[11] Byrd, Donald. Personal communication, 1998.

[12] Callan, James P. Passage-level evidence in document retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (July 1994), pp. 302–310.

[13] Cleverdon, C. W., Mills, J., and Keen, M. Factors determining the performance of indexing systems. Tech. rep., College of Aeronautics, Cranfield, 1966.

[14] Cooper, William S., Gey, Frederick C., and Dabney, D. P. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1992), pp. 198–210.

[15] Croft, W. Bruce, and Harper, David J. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation 37* (1979), 285–295.

[16] Darroch, J., and Ratcliff, D. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics 43* (1972), 1470–1480.

[17] David, Haines, and Croft, W. Bruce. Relevance feedback and inference networks. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1993), pp. 2–11.

[18] Della Pietra, Stephen, Della Pietra, Vincent, and Lafferty, John. Inducing features of random fields. *IEEE PAMI 19*, 3 (1997).

[19] Doddington, George. Topic detection and tracking evaluation. In *Proceedings of the Topic Detection and Tracking Workshop* (October 1997).

[20] Dumais, Susan T. Latent semantic indexing (lsi), trec-3 report. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3)* (1994).

[21] Elliot, Robert J., Aggoun, L., and Moore, John B. *Hidden Markov Models - Estimation and Control.* Springer-Verlag, 1995.

[22] Fuhr, Norbert. Models for retrieval with probabilistic indexing. *Information Processing and Management 25*, 1 (1989).

[23] Ghosh, M. J., Hwang, T., and Tsui, K. W. Construction of improved estimators in multiparameter estimation for discrete exponential families. *Annals of Statistics 11* (1983), 351–367.

[24] Greiff, Warren R., Croft, W. Bruce, and Turtle, Howard R. Pic matrices: A computationally tractable class of probabilistic query operators. *submitted to ACM TOIS* (1998).

[25] Harman, Donna. Overview of the fourth text retrieval conference. In *Proceedings of the 4th Text Retrieval Conference (TREC-4)* (1996), pp. 1–24.

[26] Harman, Donna. Routing results. In *Proceedings of the 4th Text Retrieval Conference (TREC-4)* (1996), pp. A53–A81.

[27] Harper, David J., and van Rijsbergen, Cornelius J. An Evaluation of Feedback in Document Retrieval Using Co-occurrence Data. *Journal of Documentation 34*, 3 (Sept 1978), 189–216.

[28] Harter, S. P. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science* (July-August 1975).

[29] Hearst, Marti. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics* (June 1994).

[30] Hearst, Marti, and Plaunt, Christian. Subtopic structuring for full-length document access. In *Proceedings of the sixteenth Annual International ACM/SIGIR Conference* (1993), pp. 59–68.

[31] Jing, Yufeng, and Croft, W. Bruce. An association thesaurus for information retrieval. Tech. Rep. 94-17, University of Massachusetts Computer Science Department, 1994.

[32] Kalt, Thomas. A new probabilistic model of text classification and retrieval. Tech. Rep. 78, CIIR, 1996.

[33] Knaus, Daniel, Mittendorf, Elke, and Shäuble, Peter. Improving a basic retrieval method by links and passage level evidence. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3)* (1994), pp. 241–246.

[34] Krovetz, Robert. Viewing morphology as an inference process. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1993), pp. 191–203.

[35] Krovetz, Robert. Word sense disambiguation for large text databases. Tech. rep., University of Massachusetts Ph.D. dissertation, 1995.

[36] Kupiec, Julian. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language 6* (1992).

[37] Lafferty, John. Personal communication, 1997.

[38] Margulis, E. L. Modeling documents with multiple poisson distributions. *Information Processing and Management 29*, 2 (1993), 215–227.

[39] Mittendorf, Elke, and Shäuble, Peter. Document and passage retrieval based on hidden markov models. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (July 1994), pp. 318–327.

[40] Moffat, Alstair, and Zobel, Justin. Self-indexing inverted files for fast text retrieval. *ACM TOIS 14*, 4 (1996), 349–379.

[41] Parzen, Emmanuel. On estimation of a probability density function and mode. *Annals of Mathematical Statistics 33* (1962).

[42] Ponte, Jay M., and Croft, W. Bruce. Useg: A retargetable word segmentation procedure for information retrieval. In *Symposium on Document Analysis and Information Retrieval '96 (SDAIR)* (1996).

[43] Ponte, Jay M., and Croft, W. Bruce. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digitial Libraries* (1997), pp. 120–129.

[44] Rabiner, Lawrence. A tutorial on hidden markov models with selected applications in speech recognition. *Proceedings of the IEEE 77* (1989), 257–285.

[45] Rabiner, Lawrence, and Juang, Biing-Hwang. An introduction to hidden markov models. *IEEE ASSP Magazine* (January 1986), 4–16.

[46] Robertson, Stephen E. The probability ranking principle in ir. *Journal of Documentation* (1977).

[47] Robertson, Stephen E., and Sparck Jones, Karen. Relevance weighting of search terms. *Journal of the American Society for Information Science 27* (1977).

[48] Robertson, Stephen E., and Walker, S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *ACM SIGIR* (1994), pp. 232–241.

[49] Rochio, Joseph J. *Relevance Feedback in Information Retrieval.* Prentice-Hall Inc., 1971, ch. 14, pp. 313–323.

[50] S., Deerwester, Dumais, Susan T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41* (1990), 391–407.

[51] Salton, Gerard. *Automatic Text Processing.* Addison Wesley, 1989.

[52] Salton, Gerard, Allan, James, and Buckley, Chris. Approaches to passage retrieval in full text information systems. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1993), pp. 49–58.

[53] Salton, Gerard, Fox, Edward A., and Wu, Harry. Extended Boolean information retrieval. *Communications of the ACM 26*, 12 (Dec. 1983), 1022–1036.

[54] Salton, Gerard, and Singhal, Amit. Automatic text theme generation and the analysis of text structure. Tech. Rep. 94-1438, Cornell University Computer Science Department, 1994.

[55] Salton, Gerard, Singhal, Amit, Buckley, Chris, and Mitra, Mandar. Automatic text decomposition using text segments and text themes. In *Proceedings of the Seventh ACM Conference on Hypertext* (March 1996), pp. 41–55.

[56] Sanderson, Mark. Word sense disambiguation and information retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1994).

[57] Silverman, Bernard W. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.

[58] Sproat, Richard, Shih, C., Gail, Willian, and Chang, N. A stochastic finite state word segmentation algorithm for chinese. *Computational Linguistics 22* (1990).

[59] Titterington, D. M., Makov, U. E., and Smith, A. F. M. *Statistical Analysis of Finite Mixture Distributions.* John Wiley and Sons, 1985.

[60] Turtle, Howard, and Croft, W. Bruce. Efficient probabilistic inference for text retrieval. In *Proceedings of RIAO 3* (1991).

[61] Turtle, Howard Robert. Inference networks for document retrieval. Tech. rep., University of Massachusetts Ph.D. dissertation, 1991.

[62] van Rijbergen, Cornelius J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation* (june 1977), 106–119.

[63] Viterbi, Andrew J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory 13*, 2 (1967), 260–269.

[64] Wayne, Charles. Topic detection and tracking introduction. In *Proceedings of the Topic Detection and Tracking Workshop* (October 1997).

[65] Wong, S. K. M., and Yao, Y. Y. A probability distribution model for information retrieval. *Information Processing and Management 25*, 1 (1989), 39–53.

[66] Wu, Z., and Tseng, G. Chinese text segmentation for text retrieval achievements and problems. *Journal for the American Society for Information Science* (October 1993).

[67] Xu, Jinxi. Solving the word mismatch problem through automatic text analysis. Tech. rep., University of Massachusetts Ph.D. dissertation, 1997.

[68] Xu, Jinxi, and Croft, W. Bruce. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (August 1996), pp. 4–11.

[69] Zelen, Martin, and Severo, Norman C. Probability functions. In *Handbook of Mathematical Functions with Formulas, Graphs, and methematical Tables: National Bureau of Standards Applied Mathematics Series No. 55.* U. S. National Bureau of Standards, 1964, ch. 26, pp. 925–996.