Inference Networks for Document Retrieval

A Dissertation Presented

by

Howard Robert Turtle

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

Doctor of Philosophy

February 1991

Department of Computer and Information Science

Inference Networks for Document Retrieval

A Dissertation Presented

by

Howard Robert Turtle

Approved as to style and content by:

W. Bruce Croft, Chair of Committee

Michael J. McGill, Member

David W. Stemple, Member

Donald L. Fisher, Outside Member

W. Richards Adrion, Department Chair
Computer and Information Science

Acknowledgments

I would like to thank Bruce Croft for his insight and his tireless support throughout this research. I would like to thank the remainder of my dissertation committee, Don Fisher, Mike McGill, and Dave Stemple for their careful review of manuscripts in preparation. Their breadth and perspective improved this dissertation in many ways.

I would like to thank OCLC Online Computer Library Center for supporting the initial phases of this research. In particular, I would like to thank Rowland Brown for his willingness to invest in the future of his company.

Abstract

INFERENCE NETWORKS FOR DOCUMENT RETRIEVAL

FEBRUARY 1991

HOWARD ROBERT TURTLE, B.A., UNIVERSITY OF WISCONSIN

M.S., UNIVERSITY OF WISCONSIN

Ph.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor W. Bruce Croft

Information retrieval is concerned with selecting documents from a collection that will be of interest to a user with a stated information need or query. Research aimed at improving the performance of retrieval systems, that is, selecting those documents most likely to match the user's information need, remains an area of considerable theoretical and practical importance.

This dissertation describes a new formal retrieval model that uses probabilistic inference networks to represent documents and information needs. Retrieval is viewed as an evidential reasoning process in which multiple sources of evidence about document and query content are combined to estimate the probability that a given document matches a query. This model generalizes several current retrieval models and provides a framework within which disparate information retrieval research results can be integrated.

To test the effectiveness of the inference network model, a retrieval system based on the model was implemented. Two test collections were built and used

to compare retrieval performance with that of conventional retrieval models. The inference network model gives substantial improvements in retrieval performance with computational costs that are comparable to those associated with conventional retrieval models and which are feasible for large collections.

æææææææææææææææææ

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we provide a brief introduction to information retrieval in order to establish a context for the research described here (Section 1.1). We then introduce four current retrieval models (Section 1.2) that we will compare with a new retrieval model developed as part of this research (Section 1.3). Finally, we provide a summary of the research conducted (Section 1.4), review our main contributions (Section 1.5), and provide an outline of the remainder of the dissertation (Section 1.6).

1.1 Overview

Information retrieval (IR) is concerned with selecting objects from a collection that may be of interest to a searcher. Objects in the collection may take many forms, they may be traditional document texts, items in a museum collection, messages in an electronic mail archive, or any form that we wish to collect for later retrieval. Objects may also exhibit complex structure in which one object is formed by combining several others (e.g., chapters may be viewed as objects that make up a book).

Collections. Information retrieval techniques that facilitate access to document collections have a history that dates back to at least the third century B.C. when the first libraries with large cataloged collections (>100,000 documents) began to appear [Hes55]. Our interest, however, is in retrieval techniques that can be applied under program control to select items from machine-readable collections. For these machine-readable collections, we generally have descriptions of the objects in the collection rather than the objects themselves (for objects that exist only in machine readable form, e.g., electronic mail messages, we may have the actual objects). These descriptions usually consist of text describing various attributes of the objects, but may also include descriptors assigned by the creator of the object or some other indexing agent (e.g., controlled vocabulary terms assigned by a human indexer or some automatically assigned classification), or used to describe relationships between objects in the collection (e.g., citations or hypertext links). Our ability to automatically interpret these descriptions, particularly the text portions, is, at present, very limited.

In most of what follows, we will assume that the objects of interest are documents. However, since our knowledge of an object is always based on a description of that object, the discussion applies to collections of other kinds of objects.

Information needs. The searcher is our most reliable source of information about whether objects are of interest. We will restrict decisions about "interest" to the context of a single "information need." The idea here is that the searcher has some more or less well defined purpose for seeking items in the collection and will make decisions about interest based on that purpose rather than in the more general context of all objects that he might find interesting. If for example, a searcher is looking for documents that deal with the computational complexity of inference networks, we expect that a document on experimental aircraft would not be judged

interesting, even if the searcher finds it to be interesting in another context. This information need is internal to the searcher and we will have only an incomplete description of the information need. This description (a query) is often expressed in natural language, but other forms are possible (e.g., sample documents, Boolean expressions). We also expect that the user's understanding of the information need will change during the search.

We will assume that the description of an information need is a description (albeit imprecise and incomplete) of characteristics that will be found in documents that match the information need. This is a major assumption that is implicit in most information retrieval research (indeed, many retrieval models essentially assume that the description of the information need is a sample document). The expected query form distinguishes information retrieval from closely related activities such as database query processing, question answering, or fact retrieval (e.g., database queries are similar to IR queries in that they represent a description of the characteristics of objects that will match the information need, but the description is entirely in terms of attributes that have a well defined semantics and no true natural language component).

Matching documents and information needs. Given a text description of an object and a text description of an information need, a human can generally (but not always) decide if the object would satisfy the information need. The kind of understanding that would allow us to make this decision under program control is, however, well beyond current natural language understanding techniques and our decision about the likelihood that a document matches an information need will be based on the fairly crude representations of the meaning or content that we can currently extract from these descriptions.

Information retrieval, then, can be seen as comprising three basic steps. Given a set of descriptions for objects in the collection and a description of an information need, we must:

1. generate a representation of the meaning or content of each object based on its description,

2. generate a representation of the meaning of the information need, and

3. compare these two representations to select those objects that are most likely to match the information need.

When we fix the details of these representations, how they are generated, and how they are compared we have defined a retrieval model.

Given this three step view of information retrieval, the major research issues fall into four broad categories:

1. What makes a good document representation? What are retrievable units and how are they organized? How can a representation be generated from a description of the document?

2. How can we represent the information need and how can we acquire this representation either from a description of the information need or through interaction with the user?

3. How can we compare representations to judge likelihood that a document matches an information need?

4. How can we evaluate the effectiveness of the retrieval process?

These categories are not independent. We cannot, for example, develop a good representation of an information need without considering the document representation and the matching process.

The primary motivation for the research described here is to improve the process of comparing representations. No current representation technique completely captures the meaning of a document or information need and there is little reason to believe that truly adequate representations will be developed in the near future. Indeed, the notion of a single representation of meaning may not be practical since the meaning of a body of text is so heavily dependent upon the context in which it is to be interpreted.

There are, however, many representation techniques that capture at least some aspects of meaning in text. The artificial intelligence (AI) community, particularly that portion interested in natural language understanding, has developed a number of techniques for representing the meaning of a text [All87]. The most successful representations make fairly specific assumptions about the way in which the text should be interpreted (e.g., as a story about one of a small number of topics) and about the kinds of questions that might be asked about the text (e.g., questions about actor's intentions). Some work has been done to adapt the natural language understanding techniques to an information retrieval setting, but there is little near-term hope that these techniques could be used to represent large document collections and arbitrary queries [Sal86].

Within the information retrieval community, a number of techniques have been developed that can represent the content of documents and information needs. These representations have a much different flavor than NLP representations. They are generally based on simple, very general, features of documents (e.g., words, citations) and represent simple relationships between features (e.g., phrases) and between documents (e.g., two documents cite the same document). The focus here is on simple, but general, representations that can be applied to most texts rather than on specialized techniques which capture more information but are applicable

only in narrow contexts. Information retrieval representations also make extensive use of the statistical properties of representation features and attempt to make use of information produced by human analysis (e.g., manual indexing) when available.

Over the last decade there has been considerable interaction between the AI and information retrieval communities; AI techniques have been adapted to an IR setting and the IR focus on "real" document collections and on thorough experimental evaluation has helped to expand the focus of AI research. The research reported here is one example of this interaction.

Given the availability of a number of representation techniques that capture some of the meaning of a document or information need, our basic premise is that decisions about which documents match an information need should make use of as many of the representation forms as practical. The remainder of this dissertation develops the theoretical framework to allow multiple representations to be combined and describes the experimental evaluation of the resulting formal model.

While our focus is on methods for comparing representations rather than on the development of new representations, we must inevitably make some assumptions about the kinds of representations that will be available and our research suggests how different kinds of representations should be organized to allow effective comparison.

1.2 Current retrieval models

A retrieval model fixes the details of the representations used for documents and information needs, describes how these are generated from available descriptions, and how they are compared. If the model has a clear theoretical basis we call it a formal retrieval model; if the model makes little or no appeal to an underlying theory

we call it *ad hoc*. We use the terms theory and model here in the mathematical or logical sense in which a theory refers to a set of axioms and inference rules that allow derivation of new theorems. A model is an embodiment of the theory in which we define the set of objects about which assertions can be made and restrict the ways in which classes of objects can interact.

Four current retrieval models are particularly relevant to this research: the Boolean, cluster-based, probabilistic, and vector-space models. We will use the measures precision and recall when describing retrieval performance. Precision is the proportion of a retrieved set that is actually relevant. Recall is the proportion of all relevant documents that are actually retrieved. These measures are discussed further in Section 2.1.

Boolean. Boolean retrieval forms the basis of most major commercial retrieval services, but is generally believed to be difficult to use and has poor precision/recall performance since the model does not rank documents. In the Boolean model we have a finite set of representation concepts or features $R = \{r_1, \ldots, r_k\}$ that can be assigned to documents. A document is simply an assignment of representation concepts and this assignment is often represented by a binary-valued vector of length k. The assignment of a representation concept r_i to a document is represented by setting the i^{th} element of the vector to *true*. All elements corresponding to features not assigned to a document are set to *false*.

An information need is described by a Boolean expression in which operands are representation concepts. Any document whose set of representation concepts represents an assignment that satisfies the Boolean expression is deemed to match the information need, all other documents do fail to match the information need. This evaluation partitions the set of documents, but provides no information about

the relative likelihood that documents within the same partition will match the information need.

Relevance in Boolean retrieval, then, is defined in terms of satisfiability of a first-order logic expression given a set of document representations as axioms. Several attempts have been made to extend the basic Boolean model to provide document ranking. The most successful of these is the Extended Boolean model that will be discussed in section 6.3.2.

Cluster-based retrieval. Cluster-based retrieval is based on the Cluster Hypothesis which asserts that similar documents will match the same information needs. Rather than comparing representations of individual documents to the representation of the information need, we first form clusters of documents using any of several clustering algorithms and similarity measures. For each cluster, we then create an "average" or representative document and compare this cluster representative to the information need to determine which clusters best match. We then retrieve the clusters that are most likely to match the information need rather than the individual documents. There are several ways to identify the clusters to be retrieved, particularly when using hierarchical clustering techniques that allow navigation of the cluster hierarchy.

Since many techniques are used to compare the query with the cluster representative, there is no single definition of relevance for cluster-based retrieval. Rather, relevance is partially defined by the model that forms the basis of the comparison. The similarity measure used to define cluster and the method used to create the cluster representatives also play a part in defining relevance since they determine which documents will be judged similar to a cluster representative that matches the information need.

Vector-space retrieval. In the vector-space model, we have a set of representation concepts or features $R = \{r_1, \ldots, r_k\}$. Documents and queries are represented as vectors of length k in which each element corresponds to a real-valued weight assigned to an element of the representations set. Several techniques have been used to compute these weights, the most common being *tf.idf* weights which are based on the frequency of a term in a single document (*tf*) and its frequency in the entire collection (*idf*). These *tf.idf* weights are discussed in more detail in Section 8.2.

Documents and queries are compared using any of several similarity functions, the most common function being the cosine of the angle between their representation vectors.

Since several techniques have been used to compute weights for the vector elements, the vector-space model has no single form of document or query representation, although all representations have a common form. Similarly, since several similarity functions have been used, relevance has no single definition.

The vector-space model is historically important since it forms the basis for a large body of retrieval research that can be traced back to the 1960's. The vector-space model has been criticized as an *ad hoc* model since there is relatively little theoretical justification for many of its variations. Some forms of the vector-space model can be shown to produce rankings that are equivalent to those produced by probabilistic models [Cro84].

Probabilistic retrieval. Probabilistic retrieval is based on the Probability Ranking Principal which asserts that the best overall retrieval effectiveness will be achieved when documents are ranked in decreasing order of probability of relevance.

There are several different probabilistic formulations which differ mainly in the way in which they estimate the probability of relevance. Using a representative model, a document d_i and an information need f_j are represented as the now

familiar vectors of length k in which each element is true if the corresponding representation concept is assigned to the document or query. If we let F represent the set of representations for information needs and D represent the set of document representations, then we can define an event space $F \times D$ and our task becomes one of determining which of these document-request pairs would be judged relevant, that is, estimating $P(R|d_i, f_j)$. We then use Bayes' theorem and a set of independence assumptions about the distribution of representation concepts in the documents and queries to derive a ranking function that computes $P(R|d_i, f_j)$ in terms of the probabilities that individual representation concepts will be assigned to relevant and non-relevant documents. Different independence assumptions lead to different forms of the model. Given estimates for these two probabilities (say, from a sample of documents judged relevant and from the entire collection), we can compute $P(R|d_i, f_j)$.

Probabilistic models are in many ways similar to the vector-space model. The main difference between the models is that probabilistic models generally have a much more rigorous theoretical base.

1.3 Inference network model

Recent retrieval research has suggested that significant improvements in retrieval performance will require techniques that, in some sense, "understand" the content of documents and queries [van86, Cro87] and can be used to infer probable relationships between documents and queries. In this view, information retrieval is an inference or evidential reasoning process in which we estimate the probability that a user's information need, expressed as one or more queries, is met given a document

as "evidence." Network representations show promise as mechanisms for inferring these kinds of relationships [CT89, CK87].

The idea that retrieval is an inference or evidential reasoning process is not new. Cooper's logical relevance [Coo71] is based on deductive relationships between representations of documents and information needs. Wilson's situational relevance [Wil73] extends this notion to incorporate inductive or uncertain inference based on the degree to which documents support information needs. The techniques required to support these kinds of inference are similar to those used in expert systems that must reason with uncertain information. A number of competing inference models have been developed for these kinds of expert systems [KL86, LK88] and several of these models can be adapted to the document retrieval task.

Network representations have been used in information retrieval since at least the early 1960's. Networks have been used to support diverse retrieval functions, including browsing [TC89], document clustering [Cro80], spreading activation search [CK87], support for multiple search strategies [CT87], and representation of user knowledge [OPC86] or document content [TS85].

In the research described here we adapt an inference network model to the retrieval task. The use of the model is intended to:

- Support the use of multiple document representation schemes. Research has shown that a given query will retrieve different documents when applied to different representations, even when the average retrieval performance achieved with each representation is the same. Katzer, for example, found little overlap in documents retrieved using seven different representations, but found that documents retrieved by multiple representations were likely to be relevant [KMT+82]. Similar results have been obtained when comparing term- with cluster-based representations [CH79] and term- with citation-based representations [FNL88].

- Allow results from different queries and query types to be combined. Given a single natural language description of an information need, different searchers will formulate different queries to represent that need and will retrieve different documents, even when average performance is the same for each searcher [MKN79, KMT+82]. Again, documents retrieved by multiple searchers are more likely to be relevant. A description of an information need can be used to generate several query representations (e.g., probabilistic, Boolean), each using a different query strategy and each capturing different aspects of the information need. These different search strategies are known to retrieve different documents for the same underlying information need [Cro87].

- Facilitate flexible matching between the terms or concepts mentioned in queries and those assigned to documents. The poor match between the vocabulary used to express queries and the vocabulary used to represent documents appears to be a major cause of poor recall [FLGD87]. Recall can be improved using domain knowledge to match query and representation concepts without significantly degrading precision.

The resulting formal retrieval model integrates several previous models in a single theoretical framework. The network model can be used to simulate probabilistic, Boolean, and cluster-based models. Moreover, retrieval results produced by these disparate models can be combined to form an overall assessment of relevance. In the network model, multiple document and query representations are treated as evidence which is combined to estimate the probability that a document satisfies a user's information need.

1.4 Research summary

In the research reported here we have developed a formal retrieval model which uses Bayesian inference networks to infer the probability that individual documents in

the collection satisfy a user's information need. This model treats multiple document representations as sources of evidence about document content and multiple query representations as sources of evidence about the information need. To simplify the presentation, we distinguish between a basic form of the model which uses the same sources of evidence as are used in simple probabilistic models (documents are represented by terms that are manually assigned or extracted from the document) and an extended model in which additional evidence forms are considered (e.g., citations, thesauri, clustering).

The network model has been implemented and evaluated using two standard test collections to compare the retrieval performance of the model with probabilistic and Boolean models. Based on the formal properties of the retrieval model and our implementation experience we developed performance bounds for building document networks and showed that queries could be evaluated efficiently.

Our original objectives for this research were:

1. Determine the retrieval performance achievable under the inference network model.

2. Compare the retrieval performance of the inference network model with that obtained with a conventional probabilistic model using the same searches run on the same databases.

3. Compare retrieval performance within the inference network model of different document representations.

4. Compare retrieval performance within the inference network model of different query representations (natural language, Boolean, and both).

5. Compare the computational performance of these networks with that of conventional probabilistic models and to characterize the costs associated with these networks as a function of network size.

These objectives led to the following hypotheses.

1. Given equivalent document representations, query forms, and assumptions about the match between indexing and query vocabularies, the inference network model will perform as well as probabilistic models.

2. The use of networks containing multiple document representations will significantly improve retrieval performance when compared to equivalent networks without the additional representations.

3. The use of multiple query formulations and search strategies will significantly improve retrieval performance when compared to equivalent networks with a single natural language query.

   Hypotheses 2 and 3 deal with the features of the basic model in which no dependencies between documents or between terms are represented.

4. The incorporation of dependencies between documents (citations and nearest neighbor links) will significantly improve retrieval performance when compared to equivalent networks that do not incorporate these additional dependencies.

5. It will be possible to build and evaluate these networks in "reasonable" time. For a collection that contains t term occurrences we will interpret "reasonable" to mean a) $O(t^2)$ time to build and preevaluate the network, b) query evaluation time relatively independent of document network size and less than 10 seconds for 90% of the queries when run on a typical micro-Vax or Sun network, and c) storage overhead that is a small constant ($c < 5$) times the original collection size.

The main results of the research are:

- The basic inference network model offers very substantial improvements in retrieval performance when compared to the best conventional retrieval models. Using equivalent evidence forms, average performance improved by 25.0%

for one test collection and 5.3% for the other when compared to baseline probabilistic retrieval. Most of this improvement results from an improved estimate of the probability that a document should be judged relevant when a term present in the query is absent from the document. These performance improvements lead us to accept Hypothesis 1.

- The network interpretation of Boolean queries performs substantially better than conventional Booleans. For the three sets of Boolean queries available with the test collections, performance improved by 57.7%, 49.1%, and 65.3% when compared to conventional Boolean retrieval. When compared to the best performance achievable with the Extended Boolean or p-norm model, performance improved by 15.1%, 7.8%, and 4.3% for the three query sets.

- The network interpretation of Boolean queries generally outperforms the original natural language versions. Booleans performed 6.6%, 0.7% and 15.7% better than the natural language versions.

- The use of Computing Reviews categories assigned to documents by authors as an additional document representation does not consistently improve performance. Performance improves significantly only when CR categories are assigned to queries by a human expert. However, the poor performance of the CR categories appears to result from problems with the assignment of CR categories in the CACM collection, so our research supports no conclusion with regard to Hypothesis 2.

- Multiple query representations significantly improve retrieval performance. When Boolean query forms are combined with the original natural language forms,

performance improves by 20.5%, 9.0%, and 17.8% for the three Boolean query sets. These results lead us to accept Hypothesis 3.

- Additional query representations can be generated automatically that will improve performance when combined with the natural language versions. These queries capture phrase structure in the natural language query and user-supplied information about the importance of terms.

- The use of additional document representations based on citation links significantly improves performance, but those based on nearest neighbor links do not. The performance improvements achieved with citations leads us to accept Hypothesis 4.

- Networks can be built and searched efficiently. Networks can be built in $O(t \lg t)$ time where t is the number of term occurrences in the collection. Average query processing time is less than one second for the test queries and average query processing time should grow logarithmically with collection size. Network files are roughly twice as large as the original source collection text and will exhibit linear or slightly sublinear growth. These performance results lead us to accept Hypothesis 5.

Implementation of the network model has also led to the development of efficient techniques for evaluating restricted classes of Bayesian inference networks. The evaluation of general Bayesian inference networks is *NP*-complete.

1.5 Research contributions

This research makes a number of contributions to information retrieval theory and has resulted in retrieval techniques of that can considerably improve the retrieval performance of conventional systems. The specific contributions of this work are:

- Very significant improvements in retrieval performance.
- Development of a formal retrieval model that generalizes probabilistic, Boolean, extended Boolean, and cluster-based retrieval models.
- Development of a theoretical framework that allows the results from several different retrieval techniques to be combined when assessing the probability that a document matches a query.
- Development of a mechanism for representing uncertain information needs. Representation of uncertainty is a very general problem that must be dealt with in many automated tasks.
- Development of techniques for representing complex information objects. Documents (and text in general) represent a particularly important class of complex object, and the techniques for integrating multiple representations of objects are of general interest.
- Development of techniques for efficient evaluation of a restricted class of probabilistic inference networks.
- This research represents a step toward more integrated information systems. The retrieval model can be integrated with a number of other types of systems (e.g., DBMS, hypertext, office automation) to improve text handling.

1.6 Outline of the dissertation

In the remainder of this dissertation we first describe the research methodology (Chapter 2) and review related work (Chapter 3). We then present the basic inference network retrieval model (Chapter 4), provide an example of its use (Chapter 5), and compare the network model to current retrieval models (Chapter 6). Finally, we describe extensions to the basic model (Chapter 7), present experimental results (Chapter 8), review implementation details (Chapter 9), and discuss conclusions and future work (Chapter 10).

Chapter 2

Experimental methodology

In this chapter we will review techniques for evaluating retrieval performance (Section 2.1) and describe the methodology used to evaluate the inference network model (Section 2.2).

2.1 Evaluation in information retrieval

The development of methods for evaluating retrieval systems and retrieval models remains an active area of research. See [Spa81] for a more complete review of evaluation for information retrieval.

Evaluations are conducted in many contexts (e.g., evaluation of an operational retrieval system, a laboratory prototype, or a paper model) with widely varying objectives (e.g., quantify retrieval effectiveness, cost effectiveness, collection quality, or user satisfaction). We will concentrate primarily on evaluation methods that can be applied to compare retrieval models in a laboratory setting. A second objective of our evaluation is to quantify the computational costs associated with retrieval techniques as they are scaled up for use in commercial settings. The issue of scalability has not received much attention in the information retrieval literature

and we make use of traditional techniques for evaluating computational complexity (see [Baa88] for a review).

The most common measures of retrieval effectiveness are *precision* and *recall*. Precision is the proportion of a retrieved set of documents that is relevant to a query. Recall is the proportion of all documents in the collection that are relevant to a query that are actually retrieved. For ranked retrieval, where there is no retrieved set, we generally compute a precision/recall pair for each relevant document in the ranking and then interpolate to find precision at standard recall points (i.e., recall= $0.1, 0.2, \ldots, 1.0$). Tables showing precision at standard recall points will be used extensively in Chapter 8. Several other measures of retrieval effectiveness have been developed but are not widely used.

2.1.1 Test collections

In order to compare the effectiveness of two retrieval models we make use of standard test collections [Sv76]. A test collection consists of:

- A set of documents – current test collections generally contain information from the original document, such as title, author, date, and an abstract. The collection may include additional information such as controlled vocabulary terms, author-assigned descriptors, and citation information. Current test collections are small, generally containing from a few hundred to a few thousand records. Documents represented in these collections are generally journal or newspaper articles.

- A set of queries – These are often actual queries submitted by users either in natural language form or in some formal query language (e.g., a Boolean expression), although artificially constructed queries are occasionally used (e.g., queries built to retrieve known documents or the text of a sample document).

- A set of relevance judgements – For each query the set of documents relevant to the query is identified. For small collections, the relevance judgements may be obtained by reviewing all documents in the collection, but for large collections relevance judgements are generally obtained by reviewing the combined results from a number of different representations of the query constructed by different searchers. Relevance judgements may be made by the query submitter or by independent domain experts. In most cases we prefer user-supplied queries and relevance judgements.

Test collections are expensive to create, primarily because developing relevance judgements for a significant number of queries is labor intensive.

When using a test collection to compare two retrieval techniques we first evaluate each query to produce a document ranking, use the relevance judgements to compute precision at standard recall points, then average the precision values over the set of queries and compare the results.

The use of test collections to evaluate retrieval performance is widespread, but there are several limitation:

- Current test collections are unrealistically small. It is not clear that techniques that are effective with test collections containing only a few thousand documents will work with commercial collections that contain several million documents. For example, a Boolean query that retrieves 3 or 4 documents from a test collection is effective since users can quickly scan the small set. The same query used with a commercial system could easily return 3,000 to 4,000 documents, far too many for the user to scan.

- Test collections generally only contain bibliographic information, while commercial systems increasingly offer the full text of newspaper and journal articles and other material types (e.g., encyclopedias or monographs). It is not clear that retrieval techniques that are effective for short bibliographic records will work well with full-text records or other material types.

- Relevance judgements are not likely to be complete for test collections of realistic size.
- Relevance judgements are subjective and are affected by a number of factors that are difficult to control. For example, the order in which documents are presented can affect whether a given document will be judged relevant.

Despite these limitations, test collections remain the most widely used method for comparing retrieval performance. New test collections are currently under development that will address the size and material type limitations [DAR90].

2.1.2 Statistical tests

Most data gathered during retrieval experiments does not lend itself well to standard statistical tests both because the quantities we wish to measure are difficult to define and because the measurements we collect do not exhibit the properties that we have come to expect in data associated with physical systems (e.g., normally distributed). [Rob81] reviews many of the definitional problems that affect experiment design. [Tag81] and [van79] review problems with statistical tests applied to retrieval data.

Of particular importance to the work reported here are tests that allow us to determine whether one set of precision/recall data is better than another. Since the data does not conform to a well-behaved distribution, we are limited to the use of non-parametric tests, principally the Sign test which counts only differences between the two samples or the Wilcoxon matched-pairs test which uses the magnitude of the differences.

2.2 Methodology

To test the effectiveness of the inference network retrieval model, inference networks were built for two test collections. The CACM test collection contains 3204 records

describing articles published in the *Communications of the ACM* from 1958 to 1979. CACM records contain author, title, abstract, and citation information as well as manually assigned keywords and Computing Review categories. The CISI test collection contains 1460 records describing highly cited information science articles published between 1969 and 1977. CISI records contain author, title, abstract, and citation information. Queries and relevance judgements are provided with each collection. A query in these collections is a natural language description of an information need. Boolean queries have been manually constructed from these natural language descriptions by the test collection providers. One set of Boolean queries is available for CISI and two sets (referred to as BL1 and BL2) are available for CACM. Summary statistics for both collections are shown in Table 2.1. See [Fox83a] for a more detailed description of the test collections and their history.

Table 2.1: Selected collection statistics

	CACM	CISI
collection size	3204	1460
unique stems	5493	5448
maximum stem frequency	1333 (algorithm)	660 (inform)
stem occurrences	117578	98304
postings	79243	71017
max within document frequency	27	27
mean within document frequency	3.7	5.2
queries	50	35

Most of the experimental work was conducted with the CACM collection since it is widely used and allows comparison of our results with a large body of previous work. Experiments with the CISI collection were carried out to validate results obtained with the CACM collection since the performance of many retrieval techniques is collection dependent. Previous studies have shown that absolute performance on

the CISI collection is low when compared to most other test collections, including CACM. CISI articles tend to be general and queries tend to be vague (e.g., "What is information science?").

While these test collections are widely used, they are far from ideal for testing the inference network model. An ideal test collection would be substantially larger than either CACM or CISI in terms of the number of documents, the length of the documents, and the number of queries. The quality of the documents in the collection would be more uniform (CACM, in particular, has a number of very short records, especially in the early part of the collection). The collection would contain a much richer set of document and query representations. Desirable document representations would include manually assigned index terms using both controlled and uncontrolled vocabularies, complete citation information, and, possibly, manually assigned links showing relationships between documents in the collection. Query representations should include the current natural language and Boolean forms plus a more conventional high-recall Boolean form prepared by a professional searcher, identification of phrases in the natural language query, and importance information for terms and phrases in both natural language and Boolean query forms.

We will use *retrieval* performance to refer to performance measured in terms of precision and recall. Retrieval performance will be described in terms of observed precision at a set of standard recall points. Precision/recall data will be presented in tabular form showing precision at ten standard recall points and the average of precision at all ten points (see Table 2.2 for an example). When two tests are being compared we show the difference as the percent change from the baseline test. While it is possible to compare performance using only average precision, this often obscures significant differences between tests. While we are interested in improving precision at all recall levels, improvements at the high precision (low recall) end of

the ranking are generally preferred since they affect documents that are most likely to be viewed by users. When computing precision/recall for a document ranking, a precision value is computed for each new relevant document retrieved. These points define a precision/recall curve. Precision values are then interpolated using pessimistic interpolation to obtain precision values at standard recall points.

Table 2.2: Effect of tie breaking strategy

	Precision (% change) – 50 queries			
Recall	random	sorted	random	sorted
10	56.8	56.8 (+0.0)	58.8	58.8 (+0.0)
20	43.6	43.4 (−0.5)	45.7	45.7 (+0.2)
30	36.2	36.1 (−0.4)	37.2	37.2 (+0.0)
40	29.3	29.1 (−0.9)	30.3	30.3 (+0.0)
50	25.5	25.2 (−1.1)	25.6	25.6 (+0.0)
60	19.1	19.2 (+0.5)	21.7	21.8 (+0.0)
70	13.9	14.0 (+1.2)	14.6	14.6 (−0.1)
80	10.7	10.7 (−0.0)	11.8	11.8 (+0.1)
90	4.5	4.5 (+0.2)	7.0	7.1 (+0.6)
100	3.0	3.0 (+0.2)	5.1	5.1 (+0.3)
average	24.3	24.2 (−0.2)	25.8	25.8 (+0.1)

Most document rankings will produce a large number of ties (especially for documents that have no terms in common with the query). The way in which these ties are broken can affect the reported retrieval performance (e.g., breaking ties in favor of relevant documents will bias the results). For the experiments reported here ties are broken by sorting tied documents in descending document identifier order. This effectively breaks ties in favor of new documents which simulates a presentation order common in commercial systems. When compared to tests in which ties were broken randomly this strategy did not significantly change retrieval performance (Table 2.2 shows the effect of random versus sorted tiebreaking for two representative tests).

Unless otherwise noted, significance tests are based on a one-tailed Sign test [Sie56] comparing the ten averaged precision values for each query set, where a 5% difference in average precision is required for two observations to be considered different. This is a conservative test, but it makes few assumptions about the distribution of observed data. As an informal rule, a difference of 5% in average precision is generally considered significant and a 10% difference is considered very significant [SB77]

We use *computational* performance to refer to the computational costs associated with building and evaluating inference networks. Computational performance will be described in terms of processor time, storage space, or bounds on representative operations.

Chapter 3

Related research

Research supporting the work reported here has been conducted in many diverse fields, including information retrieval, artificial intelligence, logic, probability theory, and database and query language models. Much of the relevant literature is discussed in the body of the dissertation, but several topics warrant separate review. In this Chapter we review research on the use of inference and inference networks when reasoning under uncertainty (section 3.1), the use of network models in information retrieval (section 3.2), and other related research (section 3.3). We conclude the Chapter by comparing the principal probabilistic inference models in current use (section 3.4).

3.1 Inference and inference networks

A number of automated inference mechanisms have been proposed, principally in the context of expert systems. Of particular interest are inference techniques that deal with uncertain information or evidence and with inference based on this evidence. Early approaches tended to be ad hoc (e.g., MYCIN's certainty factors [Sho76] or PROSPECTOR's use of probability [DHN76, DHB+78]). The development of more

formal techniques has led to heated debate among several competing schools. We will not review the debate in detail (see [KL86, LK88, SP90] for surveys), but three main approaches have emerged. The first approach relies on symbolic reasoning [Coh85, Fox86, Doy79] in which degrees of certainty are encoded using a discrete set of values (certain, somewhat certain, ...) that are then used with a deductive reasoning system. The second approach uses fuzzy set theory [Zad83, Zad86a]. The third approach is based on probabilistic methods. While the use of probabilistic methods is gaining acceptance, the appropriateness of these techniques has been hotly debated. An essay by Cheeseman [Che88] and the follow up discussion in a special issue of Computational Intelligence provides a good summary of the issues.

Two main probabilistic approaches are in use. The first uses conventional probabilistic methods [Pea88, LS88, AOJJ89] and the second uses the Dempster-Shafer theory of evidence [Dem68, Sha76, Sha87a, Zad86b]. The two approaches are similar; the Dempster-Shafer approach represents an attempt to generalize Bayesian methods in order to cope with the fact that a complete probability distribution is rarely available. The February 1987 issue of Statistical Science includes a debate between proponents of the two approaches [Sha87b, Lin87, Spi87]. Spiegelhalter [Spi86] compares the Dempster-Shafer and Bayesian methods. Pearl [Pea88] compares Dempster-Shafer with Bayesian inference networks and describes conditions under which they are equivalent. Gordon and Shortliffe [GS84] compare Dempster-Shafer with MYCIN's certainty factors. Lowrance and Garvey [GLF81, LGS86] describe the use of Dempster-Shafer for evidential reasoning. Spiegelhalter and Knill-Jones [SKJ84] and Andersen *et al* [AOJJ89] describe the use of Bayesian methods in clinical applications.

Other probabilistic approaches have been developed but have not been as widely accepted. Nilsson's probabilistic logic [Nil84, Nil86] represents an alternative method

of dealing with incomplete probability models by estimating bounds for probabilities rather than point estimates. Quinlan's INFERNO [Qui83] incorporates these kinds of probability bounds.

The same three inference approaches (symbolic, fuzzy set, probabilistic) are evident in information retrieval research. Symbolic approaches include those based on Boolean logic and on relational algebra [Bla88] or calculus. Retrieval models based on fuzzy sets have been proposed [Boo85, Rad79, KB84, Zad86a] and a wide variety of probabilistic models have been explored [MK60, RS76, CM78, van79, Fuh89a] which exploit the statistical properties of text in an attempt to improve performance.

Van Rijsbergen [van86, van89] discusses the nature of inference in information retrieval and has proposed the use of non-classical logics for determining the degree to which a document implies or matches a query. Croft [Cro87, CLCW89] has developed the notion of plausible inference in information retrieval and suggested that multiple sources of evidence should be combined to infer the probability that a document matches a query. The research described here extends the work of vanRijsbergen and Croft and provides a formal computational model of inference in document retrieval.

The use of networks to represent the inference process is common, even when the network is not an essential part of the inference model or its implementation (e.g., Spiegelhalter's [Spi86] description of MYCIN). In other cases, inference models that do not depend on an underlying network have been adapted for use in an inference network [GLF81, GS84, LGS86]. Some of the more recent models depend on a network model and rely heavily on the graph theoretic aspects of the network to demonstrate properties of the inference model. Pearl relies heavily on graph properties in his axiomatic development of Bayesian inference networks [PP85, PV87] as do

Lauritzen and Spiegelhalter in their development of the properties of causal nets in MUNIN [LS88]. The network models also allow use of the extensive complexity results for graph operations when reasoning about the complexity of inference procedures.

While these inference networks were developed for use in expert systems, they are very similar to the influence diagrams [Sha86] of operations research and to techniques developed for pedigree analysis [CTS78], legal reasoning [SM82], and intelligence analysis [Sch87].

3.2 Network models in information retrieval

Graph and network structures have been widely used in information retrieval. Salton [Sal68] describes early use of tree and graph models in information retrieval and describes implementation of many of the basic structures used in retrieval systems (e.g., inverted files, dictionaries) in graph theoretic terms. Salton and McGill [SM83] and van Rijsbergen [van79] provide more current introductions to common retrieval structures, many of which are graph or network based. Other uses of networks in information retrieval can be loosely categorized as support for clustering, rule-based inference, structure matching, browsing, spreading activation, and connectionist approaches. This categorization is rough and individual systems could be assigned to multiple categories.

Clustering. Networks arise naturally in the representation of document and term clusters. Croft [Cro81] describes a retrieval model incorporating document and term clusters. Croft and Parenty [CP85] compare a cluster based network representation with a conventional database implementation. Willett [Wil88] re-

views document clustering techniques and Sparck Jones [Spa71, Spa74] reviews term clustering techniques.

Rule-based inference. In RUBRIC [TSMD83, TS85], Tong represents queries as a set of rules in an evaluation tree that specifies how individual document features can be combined to estimate the certainty that a document matches the query. One of the objectives of the RUBRIC design was to allow comparison of different uncertainty calculi [TAAC86] and RUBRIC has recently been reformulated to use inference networks [FCAT90]. Rule-based inference using network structures has been used with thesaurus information to improve the match between document and query vocabularies [CT87, Sho85]. Semantic networks have also been used to represent thesaurus-like information [SSG89, MC87].

Structure matching. Structure matching forms the basis of most retrieval techniques based on semantic networks. Early work by Salton [Sal68] describes the use of graphs to represent the syntactic structure of text and graph matching to identify similar text content. Lewis, Croft, and Bhandaru [LCB89] discuss the use of frame-based networks produced by natural language parsers to represent documents and queries and network matching functions that can be used for retrieval. Structure matching also underlies the simpler network structures used by Belkin *et al* to represent anomalous states of knowledge (ASK) [BOB82a, BOB82b, BK86, OPC86] regarding document and query content.

Browsing. When networks are used to represent documents and indexing information, browsing can be used to help users locate relevant material. Browsing is common in thesaurus systems. Oddy's THOMAS system [Odd77] uses browsing in a simple network of documents and terms to build a model of the user's information need. Croft and Thompson use browsing in a more complex network as one search strategy in I^3R [CT87].

Browsing is an important technique for accessing text in hypertext networks. Croft and Turtle [CT89] and Frisse and Cousins [FC89] describe retrieval models for hypertext networks. Coombs [Coo90a] reviews hypertext retrieval research and describes an operational system.

Spreading activation. Spreading activation is a search technique in which a network representation of a document collection is used to retrieve documents that are "similar" to a query. The query is used to activate a set of nodes in a representation network which, in turn, activate neighboring nodes. Halting conditions and weighting functions vary, but the pattern of activation is used to rank documents for presentation to the user. Jones and Furnas [JF87] present a representative spreading activation model which is compared to conventional retrieval models by Salton [SB88]. Croft [CLCW89] used spreading activation in a network based on document clustering. Cohen and Kjeldson [CK87] used spreading activation in a more complex representation network with typed edges.

Connectionist approaches. Connectionist approaches are similar to spreading activation. They differ in that the connectionist links do not have a clear semantic interpretation (they simply characterize the "association" between network nodes) and the weights associated with links are learned from training samples or user guidance. Croft and Thompson [CT84] use a connectionist network in an attempt to learn to select a query strategy. Brachman and McGuiness [BM88] use a connectionist approach to retrieve facts from a knowledge base on programming languages. Other connectionist approaches to information retrieval are described by Belew [Bel89] and Kwok [Kwo89]. Lewis [Lew90] further explores the relationship between information retrieval and machine learning.

The connectionist retrieval approaches that have been reported use simple networks with no hidden units and thus learn simple linear discriminant functions.

The inference network model supports dependence structures that require nonlinear functions and require more complex learning strategies. [Pea88, CF89, FCAT90] and [CL68] discuss learning strategies that are applicable to inference networks.

3.3 Other related research

Two additional areas of information retrieval research bear directly on the current work – traditional strategies for term weighting that can be used as the basis for estimating beliefs in the network and the use of citations to improve retrieval performance.

Term weighting. When selecting terms to be used to represent the content of documents and queries, we can simply use the presence or absence of a term (*feature* would probably be better here since a "term" in this context might be a word, phrase, controlled vocabulary element, or other descriptor) to characterize the relationship between the term and the document or query, or we can develop an estimate of the term's importance or quality in the form of a term or feature weight. These term weights have been studied extensively in information retrieval (see [SB77, van79, SB87, SM83] for reviews) and a number of theoretical models have been developed to motivate the assignment and interpretation of these weights [CM78, RS76, BS75, SYY75].

In general, effective term weighting strategies have two components. The first is an estimate of how well a term can discriminate documents in the collection. In this regard, terms that occur in a high percentage of the documents in a collection are poor discriminators and those that occur rarely are good discriminators. Estimates for this component are generally based on a term's inverse document frequency (*idf*) which will be discussed in Chapter 8.

The second component is an estimate of how important a term is in describing the content of an individual document or query. Estimates for this component are generally based on the frequency with which a term occurs in the document or query; high frequency terms are generally thought to be more important descriptors.

These two components can be combined in several ways, may be normalized for document or query length, and are often scaled if they are interpreted as probability estimates (see [SB87] for a comparison of several weighting strategies).

Some work on estimating weights for phrases rather than single terms has been done [Fag87, LC90], but our understanding of how phrases can be used to improve retrieval performance is still limited.

Citations. The use of citation information in an attempt to improve retrieval performance dates back to at least the early 1960's [Sal63]. Document similarity measures based on citation information have been developed. *Bibliographic coupling* describes the degree to which the citations found in two documents overlap. *Co-citation*-based measures are based on the frequency with which a pair of documents are cited together in a third document[1]. It has been shown that documents with high citation-based similarity scores also have similar index terms [Kes65, Sma73].

Citations can be used navigationally, where we first find a relevant document and then use the citations to find other potentially relevant documents, or they can be treated as indexing features – each citation in a document is treated as if it were an index term which can be used in a query. Several studies have found that searches using natural language terms retrieve different documents for the same information need [Hur82, PF85, GWDS86]. In a recent study comparing the performance of term- and citation-based searches, Pao and Worthen [PW89] found that citation

[1]It is sometimes useful to distinguish between a list of citations and a bibliography which may include related but uncited documents. We will largely ignore this distinction.

searches found a significant number of documents (14% of all relevant documents) that were not found using natural language terms. They also found that the set of documents retrieved by both searches was small (12% of all relevant documents) but that documents in this set were highly relevant.

3.4 Probabilistic inference models

Probabilistic methods are among the most effective tools known for improving retrieval effectiveness and information retrieval research has produced a substantial body of knowledge about the statistical properties of text. Since we wish to build on this research, the two main inference models based on probabilistic methods are of particular interest: Bayesian inference networks and the Dempster-Shafer theory of evidence.

A Bayesian inference network is a directed, acyclic dependency graph in which nodes represent propositional variables or constants and edges represent dependence relations between propositions. If a proposition represented by a node p "causes" or implies the proposition represented by node q, we draw a directed edge from p to q. The node q contains a matrix (a *link* matrix[2]) that specifies $P(q|p)$ for all possible values of the two variables. When a node has multiple parents, the matrix specifies the dependence of that node on the set of parents (π_q) and characterizes the dependence relationship between that node and all nodes representing its potential causes. Given a set of prior probabilities for the roots of the DAG, these networks can be used to compute the probability or degree of belief associated with all remaining nodes.

[2]The link matrix is actually a link *tensor*. See Appendix .

Different restrictions on the topology of the network and assumptions about the way in which the connected nodes interact lead to different schemes for combining probabilities. In general, these schemes have two components which operate independently: a *predictive* component in which parent nodes provide support for their children (the degree to which we believe a proposition depends on the degree to which we believe the propositions that might cause it), and a *diagnostic* component in which children provide support for their parents (if our belief in a proposition increases or decreases, so does our belief in its potential causes). The propagation of probabilities through the net can be done using information passed between adjacent nodes.

While not originally cast as a network model, the Dempster-Shafer theory of evidence can be used as an alternative method for evaluating these kinds of probabilistic inference networks. Rather than computing the belief associated with a query given a set of evidence, we can view Dempster-Shafer as computing the probability that the evidence would allow us to prove the query. The degree of support parameters associated with the arcs joining nodes are not interpreted as conditional probabilities, but as assertions that the parent node provides support for the child (is *active*) for some proportion p of the time and does not support the child for the remainder of the time. For an *and*-combination we compute the proportion of the time that all incoming arcs are active. For an *or*-combination we compute the proportion of the time that at least one parent node is active. To compute the provability of the query given a document, we examine all paths leading from the document to the query and compute the proportion of time that all of the arcs on at least one proof path are active. Given the structure of these networks, this computation can be done using series parallel reduction of the subgraph joining the document and query in time proportional to the number of arcs in the subgraph.

In general, the Bayesian and Dempster-Shafer models are different and can lead to different results. Under the assumption of disjunctive rule interaction (so called "noisy-OR") and the interpretation of an arc from a to b as $P(b|a) = p$ and $P(b|\neg a) = 0$, the Bayesian and Dempster-Shafer models will produce similar results [Pea88, page 446]. The document retrieval inference networks described here are based on the Bayesian inference network model.

The use of Bayesian inference networks for information retrieval represents an extension of probability-based retrieval research dating from the early 1960's. The use of these networks generalizes existing probabilistic models and allows integration of several sources of knowledge in a single framework.

Chapter 4

Basic Model

In this chapter we introduce the inference network model (Sections 4.1 and 4.2) and show how it is used to identify documents that are most likely to match a user's information need (Section 4.3). We then discuss "causation" in Bayesian networks (Section 4.4) and describe the canonical link matrix forms that are used in the model (Section 4.5).

The basic document retrieval inference network, shown in Figure 4.1, consists of two component networks: a document network and a query network. The document network represents the document collection using a variety of document representation schemes. The document network is built once for a given collection and its structure does not change during query processing. The query network consists of a single node which represents the user's information need and one or more query representations which express that information need. A query network is built for each information need and is modified during query processing as existing queries are refined or new queries are added in an attempt to better characterize the information need. The document and query networks are joined by links between representation concepts and query concepts. All nodes in the inference network are binary-valued and take on values from the set $\{false, true\}$.

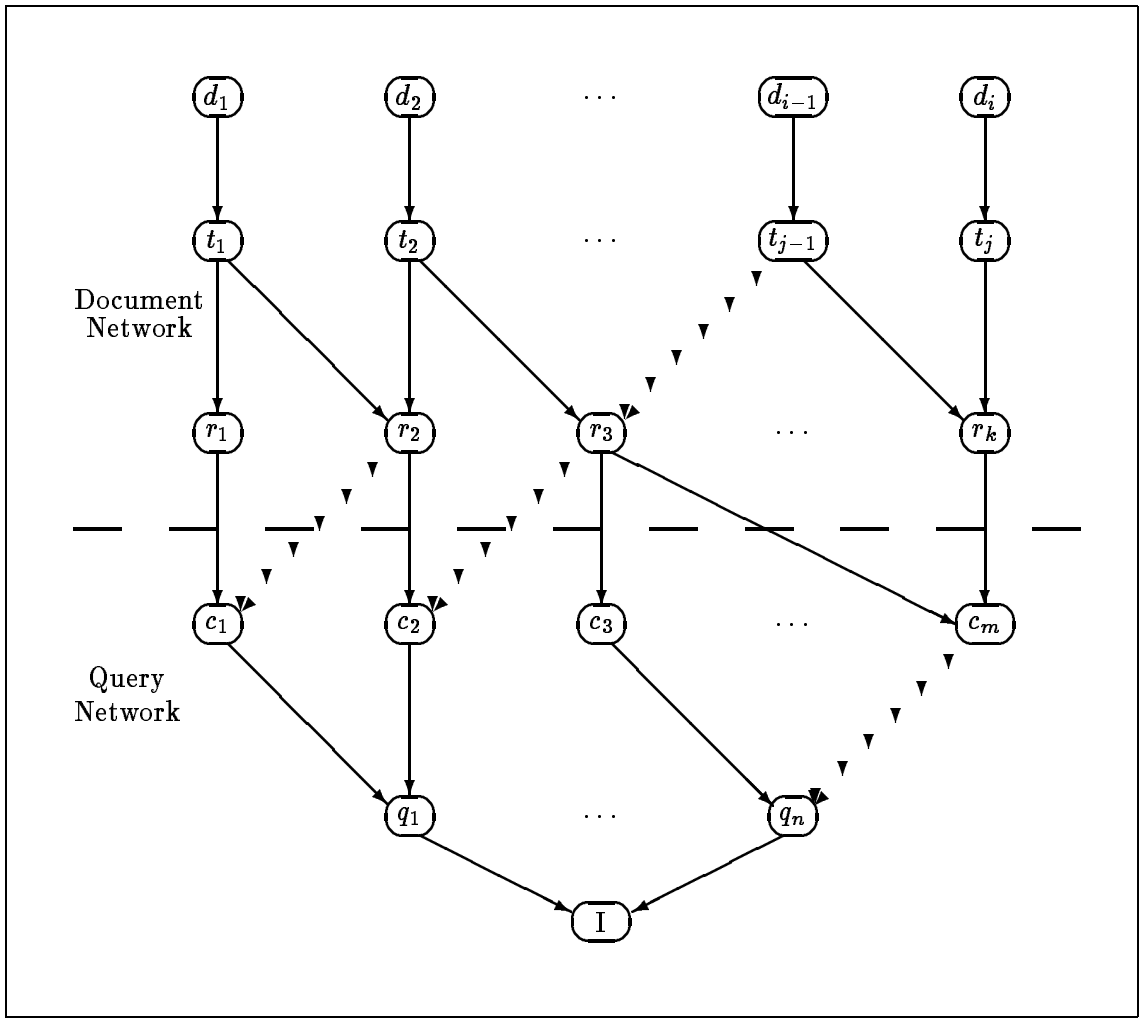

Figure 4.1: Basic document inference network

4.1 Document network

The document network consists of document nodes (d_i's), text representation nodes (t_j's), and concept representation nodes (r_k's). If we let D be the set of documents, T be the set of text representations, and R be the set of representation concepts, where the cardinality of these sets is n_d, n_t, and n_r, respectively, then the event space represented by the document network is $E_d = D \times T \times R$. Since all propositions are binary-valued, the size of the event space is $2^{n_c} \cdot 2^{n_t} \cdot 2^{n_r}$.

Each document node represents an actual document in the collection. A document node corresponds to the event that a specific document has been observed. The form of the document represented depends on the collection and its intended use, but we will assume that a document is a well defined object and will focus on traditional document types (e.g., monographs, journal articles, office documents, ...).

Document nodes correspond to abstract documents rather than their physical representations. A text representation node or text node corresponds to a specific text representation of a document. A text node corresponds to the event that a text representation has been observed. We will focus here on traditional document texts, but one can easily imagine other content types for documents (e.g., figures) and multi-media documents might have several content representations (e.g., audio or video). In these cases, a single document might have multiple physical representations. Similarly, a single text content might be shared by more than one document. While this sharing is rare (an example would be a journal article that appears in both a serial issue and in a reprint collection) and is not generally represented in current retrieval models, it is common in hypertext systems. For clarity, we will only consider text representations and will assume a one-to-one correspondence between

documents and texts. The dependence of a text upon the document is represented in the network by an arc from the document node to the text node.

The content representation nodes or representation nodes can be divided into several subsets, each corresponding to a single representation technique that has been applied to the document texts. For example, if a collection has been indexed using automatic phrase extraction and manually assigned index terms, then the set of representation nodes will consist of two distinct subsets or content representation types with disjoint domains. Thus, if the phrase "information retrieval" has been extracted and "information retrieval" has been manually assigned as an index term, then two representation nodes with distinct meanings will be created. One corresponds to the event that "information retrieval" has been automatically extracted from a subset of the collection, the second corresponds to the event that "information retrieval" has been manually assigned to a (presumably distinct) subset of the collection. We represent the assignment of a specific representation concept to a document by a directed arc to the representation node from each text node corresponding to a document to which the concept has been assigned. For now we assume that the presence or absence of a link corresponds to a binary assigned/not assigned distinction, that is, there are no partial or weighted assignments.

In principle, the number of representation schemes is unlimited. In addition to phrase extraction and manually assigned terms we would expect representations based on natural language processing and automatic keyword extraction. Refinements that can be applied to multiple representations (e.g., thesauri, term clustering, or inference rules) will be discussed in section 7. For any real document collection, however, the number of representations used will be fixed and relatively small. The potential domain of each representation scheme may also be unlimited, but the actual number of primitive representation concepts defined for a given

collection is fixed by the collection. The domain for most automated representation schemes is generally bounded by some function of the collection size (e.g., the number of keywords cannot exceed the number of words in a collection). For manual representation schemes the domain size is limited by the number of documents and the amount of time a human expert can invest to analyze each document.

The basic document network shown in Figure 4.1 is a simple three level directed acyclic graph (DAG) in which document nodes are roots, text nodes are interior nodes, and representation nodes are leaves. Document nodes have exactly one text node as a child and each text node has one or more representation nodes as children.

Each document node has a prior probability associated with it that describes the probability of observing that document; this prior probability will generally be set to 1/(collection size) and will be small for reasonable collection sizes. Each text node contains a specification of its dependence upon its parent; by assumption, this dependence is complete, a text node is observed ($t_i = true$) exactly when its parent document is observed ($d_i = true$).

Each representation node contains a specification of the conditional probability associated with the node given its set of parent text nodes. This specification incorporates the effect of any indexing weights (e.g., term frequency for each parent text) or term weights (e.g., inverse document frequency) associated with the representation concept. While, in principle, this would require $O(2^n)$ space for a node with n parents, in practice we will generally use canonical representations that will allow us to compute the required conditional probabilities when needed. These canonical schemes are described in section 4.5 and require $O(n)$ space if we need to weight the contribution of each parent or $O(1)$ space if parents are to be treated uniformly.

4.2 Query network

The query network is an "inverted" DAG with a single leaf that corresponds to the event that an information need is met and multiple roots that correspond to the concepts that express the information need. As shown in Figure 4.1, a set of intermediate query nodes may also be used in cases where multiple query representations are used to express the information need. These nodes are a representation convenience; it is always possible to eliminate them by increasing the complexity of the distribution specified at the node representing the information need.

If we let C represent the set of query concepts and Q represent the set of queries where n_c and n_q are the cardinalities of these sets, then the event space represented by the query network is $E_q = C \times Q \times I$. Since we can always eliminate query nodes, $|E_q| \leq 2^{n_c+1}$. The event space represented by the entire inference network is then $E_d \times E_q$.

In general, the user's information need is internal to the user and is not precisely understood. We attempt to make the meaning of an information need explicit by expressing it in the form of one or more queries that have a formal interpretation. It is unlikely that any of these queries will correspond precisely to the information need, but some will better characterize the information need than others, and several query representations taken together may be a better representation of the information need than any of the individual queries.

The roots of the query network are query concepts, the primitive concepts used to express the information need. A single query concept node may have several representation concept nodes as parents. A query concept node contains a specification of the probabilistic dependence of the query concept on its set of parent representation concepts. The query concept nodes define the mapping

between the concepts used to represent the document collection and the concepts that make up the queries. In the simplest case, the query concepts are constrained to be the same as the representation concepts and each query concept has exactly one parent representation node. In a slightly more complex example, the query concept "information retrieval" may have as parents both the node corresponding to "information retrieval" as a phrase and the node corresponding to "information retrieval" as a manually assigned term.

As we add new forms of content representation to the document network and allow the use of query concepts that do not explicitly appear in any document representation, the number of parents associated with a single query concept will tend to increase. In many ways, a query concept is similar to a representation concept that is derived from other representation concepts (see section 7.2.3 for a discussion of derived representation concepts) and in some cases it will be useful to "promote" a query concept to a representation concept. For example, suppose that a researcher is looking for information on a recently developed process that is unlikely to be explicitly identified in any existing representation scheme. The researcher is sufficiently motivated, however, to work with the retrieval system to describe how this new concept might be inferred from other representation concepts. If this new concept definition is of general interest, it can be added to the collection of representation concepts. The process of defining new representation concepts is similar to that used in RUBRIC [TSMD83, TS85] where a user might add a rule which asserts that the concept "car bomb" should be inferred with some level of certainty if the term "car" and "bomb" occur in the same sentence. The RUBRIC approach differs in that all representation concepts are manually defined, whereas most representation concepts in an inference network are created automatically.

The attachment of the query concept nodes to the document network has no effect on the basic structure of the document network. None of the existing links need change and none of the conditional probability specifications stored in the nodes are modified.

A query node represents a distinct query representation and corresponds to the event that the query representation is satisfied. Each query node contains a specification of the dependence of the query on the query concepts it contains. The content of the link matrices that contain the conditional probabilities is discussed further in section 4.5, but it is worth noting that the form of the link matrix is largely determined by the query type; a link matrix simulating a Boolean query is much different than a matrix simulating a probabilistic or weighted query.

Multiple query representations can be obtained from many sources. It is possible that the user might provide more than one form (e.g., a natural language description and a sample document), but it is more likely that additional forms will be generated automatically based on the original natural language query or using information obtained by an intelligent interface. In cases where a search intermediary is used, we may have multiple human-generated query representations. In Chapter 8 we describe some initial experiments in which new query representations are generated using natural language processing (NLP) techniques and using additional information that users could supply.

The single leaf representing the information need corresponds to the event that an information need is met. In general, we cannot predict with certainty whether a user's information need will be met by an arbitrary document collection. The query network is intended to capture the way in which meeting the user's information need depends on documents and their representations. Moreover, the query network is intended to allow us to combine information from multiple document representations

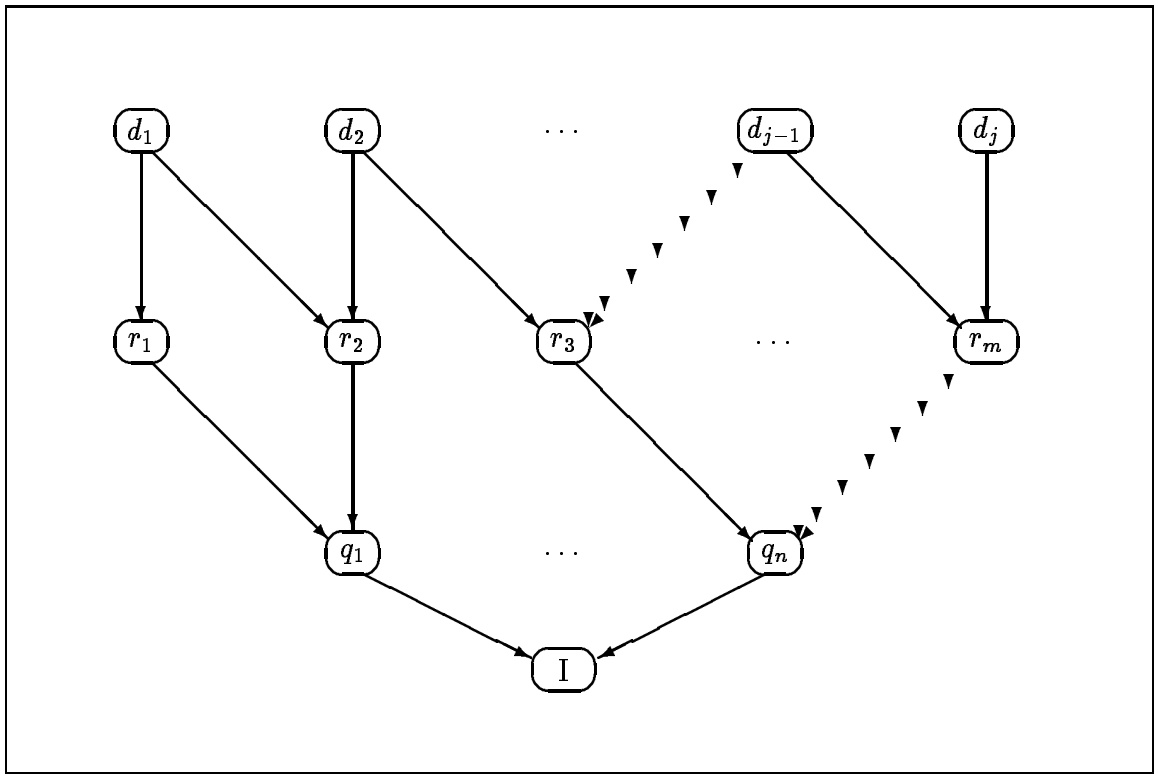

Figure 4.2: Simplified inference network

and to combine queries of different types to form a single, formally justified estimate of the probability that the user's information need is met. If the inference network correctly characterizes the dependence of the information need on the collection, the computed probability provides a good estimate.

We will often use a simplified form of the basic model of Figure 4.1 in which we assume a one-to-one correspondence between document nodes and text nodes and between representation concept and query concept nodes. Under these assumptions, the network of Figure 4.1 can be reduced to the network shown in Figure 4.2.

4.3 Use of the inference network

The inference network we have described is intended to capture all of the significant probabilistic dependencies among the variables represented by nodes in the document and query networks. Given the prior probabilities associated with the documents (roots) and the conditional probabilities associated with the interior nodes, we can compute the posterior probability or belief associated with each node in the network. Further, if the value of any variable represented in the network becomes known we can use the network to recompute the probabilities associated with all remaining nodes based on this "evidence."

The network, taken as a whole, represents the dependence of a user's information need on the documents in a collection where the dependence is mediated by document and query representations. When the query network is first built and attached to the document network we compute the belief associated with each node in the query network. The initial value at the node representing the information need is the probability that the information need is met given that no specific document in the collection has been observed and all documents are equally likely (or unlikely). If we now observe a single document d_i and attach evidence to the network asserting $d_i = \mathit{true}$ with all remaining document nodes set to *false* (referred to as *instantiating* d_i), we can compute a new belief for every node in the network given $d_i = \mathit{true}$. In particular, we can compute the probability that the information need is met given that d_i has been observed in the collection. We can now remove this evidence and instead assert that some $d_j, i \neq j$ has been observed. By repeating this process we can compute the probability that the information need is met given each document in the collection and rank the documents accordingly.

In principle, we need not consider each document in isolation but could look for the subset of documents which produce the highest probability that the information need is met. While a general solution to this best-subset problem is intractable, in some cases good heuristic approximations are possible. Best-subset rankings have been considered in IR [Sti75, Boo89], and similar problems arise in pattern recognition, medical diagnosis, and truth-maintenance systems. See [Pea88] for a discussion of the best-subset or belief revision problem in Bayesian networks. At present, we consider only documents in isolation since the approach is computationally simpler. As will be discussed in Chapter 9, this simplification is an important factor in reducing the exponential complexity of network evaluation. Note that any retrieval model that produces document rankings that are consistent with conventional formulations of the Probability Ranking Principle [Rob77] must also consider documents in isolation.

The document network is built once for a given collection. Given one or more queries, we then build a query network that attempts to characterize the dependence of the information need on the collection. If the ranking produced by the initial query network is inadequate, we must add additional information to the query network or refine its structure to better characterize the meaning of the existing queries. This feedback process is quite similar to that used in current retrieval systems. The process of translating queries into network forms is discussed further in Chapter 6.

4.4 Causation in Bayesian inference networks

The notion of causation, that one random variable can be perceived as causing another, is fundamental to Bayesian inference networks. By drawing an arc from node a to node b we are asserting that a in some sense causes b. If a is observed,

then our belief in b is fixed by that observation (assuming b has no other parents). If we later observe b to have a value that conflicts with our computed belief we suspect that either the conditional probability $P(b|a)$ is incorrect or that the topology is wrong (either b has causes we haven't recognized or a does not, in fact, cause b). If, however, we first observe b then our belief in a changes because a is a potential explanation for b, that is, the observation of b constitutes evidence confirming or disconfirming a.

While in many cases the direction of causation is clear (e.g., most instances of physical causation), in many others it is difficult to distinguish between causal and evidential support. For example, our network in Figure 4.1 asserts that our belief in a set of query concepts causes our belief in the query that contains them. We could also have argued that our belief that the query is a representation of the information need causes our belief that the query concepts are useful. In this case we view the query concepts as evidence that supports our belief in the query.

In our network in Figure 4.1 we assert that the observation of a document (or a set of documents) causes our belief in a text representation, which causes our belief in a set of representation concepts, which in turn cause belief in a set of query concepts, which cause our belief in a set of queries, which finally cause our belief that the document supports the information need. In fact, there are (at least) two other topologies that have some intuitive appeal. In the first, we simply invert the entire network. This structure asserts that the information need causes our belief in the queries, which cause our belief in the query concepts they contain. While this chain of causation is at least plausible, the next step in which query concepts cause our belief in representation concepts, is not very appealing. Since documents and their representations have an existence independent of any query network, the query

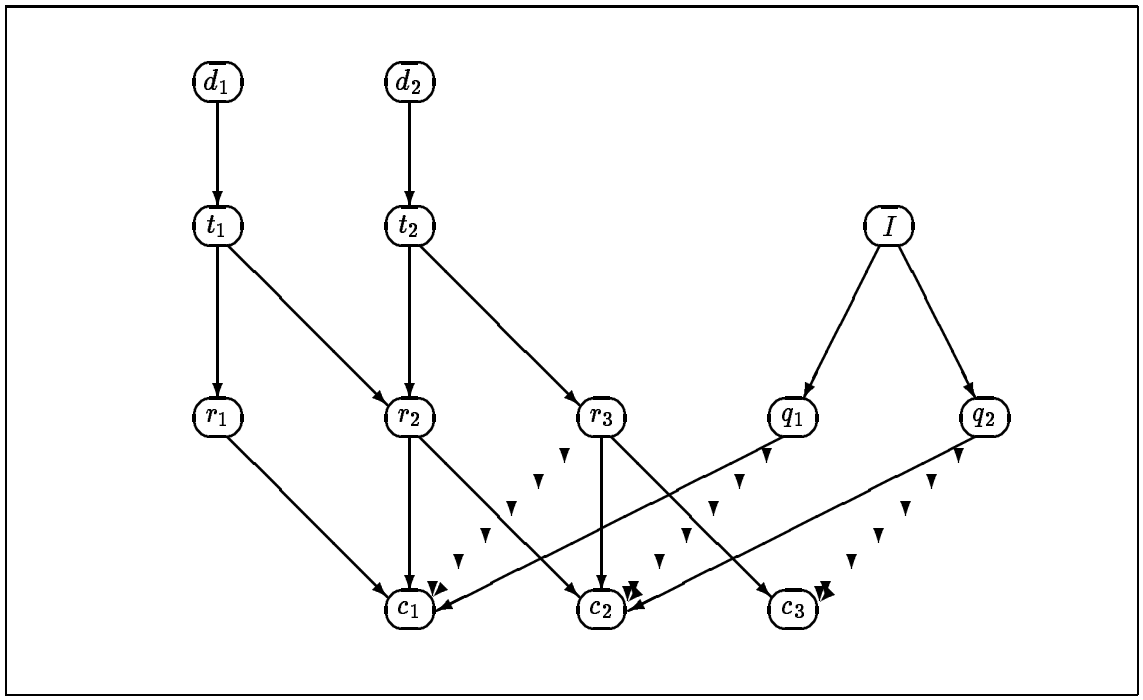

Figure 4.3: Mixed document inference network

concepts cannot cause the representation concepts; our belief that a representation concept is assigned to a set of documents is not altered by the processing of a query.

In some retrieval models (e.g., document space modification [YM88, FB90]) queries are used in an attempt to learn better document representations, which gives rise to an indirect causal relationship between queries and representation concepts that is not represented in the inference network. In these models the queries do not directly influence our belief in representation concepts, they are simply used to improve our characterization of the dependence of representation concepts on document texts.

A second variation, shown in Figure 4.3 asserts that the documents are the ultimate causes of the document network and the information need is the ultimate cause of the query network. The two nets are connected by links establishing the

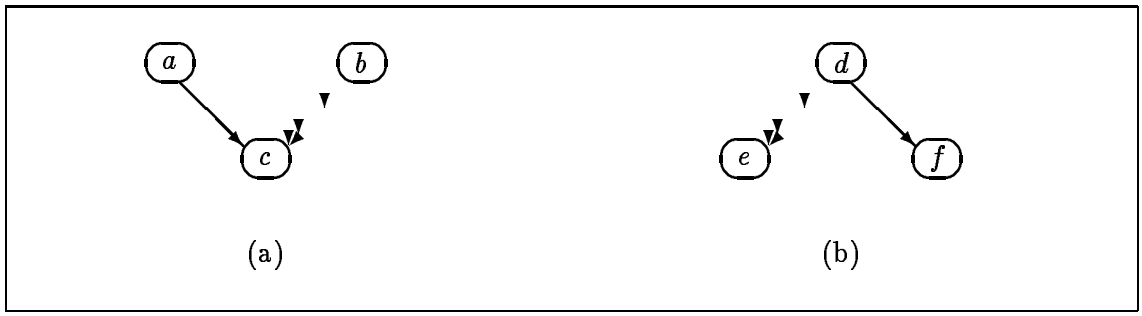

Figure 4.4: Basic causal topologies

dependence of query concepts on representation concepts. To see why this network does not capture our intuition about the relationships among variables we need to look more closely at how variables interact in these causal structures.

In Figure 4.4a, two nodes a and b are potential causes of c. If c has not been observed, a and b are independent. Changes in our belief in a will affect our belief in c but will have no effect on our belief in b. This clearly does not model the desired behavior in Figure 4.3 where we would like our beliefs about query concepts induced by the document network to propagate up the query network to affect our belief in the information need. If, in Figure 4.4a we observe c, then a and b become dependent since they are both potential causes for c. In effect, they are competing explanations for c. If we now observe a to be true, our belief in b diminishes because a fully accounts for our observation of c. This behavior also fails to capture the desired relationship between the document and query networks in Figure 4.3; we would expect that when belief in the representation concepts supporting the query concepts increases, our belief in the information need would also increase. Clearly, the two networks are not competing to support the query concepts.

Figure 4.4b shows the second case of interest. In this network a single node d causes both e and f. In the absence of any observation of d, e and f are dependent.

If e is observed to be true, this evidence raises belief in d which in turn raises belief in f. In the absence of any evidence at d, any evidence collected at one of its children directly affects all children. Once d is observed, however, e and f become independent; further evidence gathered at one child will not affect belief in d (since its value is known) and can therefore have no impact on its siblings.

The interaction between the variables in these two network fragments helps to clarify the nature of "causation" in Bayesian inference nets. Two causes sharing a common consequence are independent until the consequence is observed, thereafter they compete. Two consequences of a common cause are dependent and share support until the cause is observed. Thereafter, the consequences are independent.

4.5 Link matrix forms

For all non-root nodes in the inference network we must estimate the probability that a node takes on a value given any set of values for its parent nodes. If a node a has a set of parents $\pi_a = \{p_1, \ldots, p_n\}$, we must estimate $P(a|p_1, \ldots, p_n)$.

The most direct way to encode our estimate is as a link matrix (recall, however, that a link matrix is actually a link tensor). Since we are dealing with binary valued propositions, this tensor can be represented by a matrix of size 2×2^n for a node with n parents and specifies the probability that a takes the value $a = \mathit{true}$ or $a = \mathit{false}$ for all combinations of parent values. The update procedures for Bayesian networks then use the probabilities provided by the set of parents to condition over the link matrix values to compute the predictive component of our belief in a or $P(a = \mathit{true})$. Similarly, the link matrix is used to provide diagnostic information to the set of parents based on our belief in a. As discussed earlier, encoding our estimates in link matrix form is practical only for nodes with a small set of parents,

so our estimation task has two parts: how do we estimate the dependence of a node on its set of parents and how do we encode these estimates in a usable form?

In this section we will define canonical link matrix forms that are useful for retrieval networks and review the time and space complexity associated with evaluating these canonical forms. By a canonical form we mean that, given an ordering on a set of n parents, we can compute the link matrix value $L[i,j], i \in \{0,1\}, 0 \leq j < 2^n$ given the parent index corresponding to p_j. When writing a link matrix we use the row number to index values assumed by the child node and use a binary representation of the column number to index the values of the parents. We use the high order bit of the column number to index the first parent's values, the second most high order for the second parent, and so on. The three-parent example of the next section illustrates this notation.

4.5.1 Canonical link matrix forms

We will describe five canonical link matrix forms. Three of these forms implement the Boolean operators *and*, *or*, and *not*; the remaining two forms implement weighted-sums that are used for probabilistic retrieval. A number of other forms are possible.

For illustration, we will assume that a node Q has three parents A, B, and C (Figure 4.5) and that

$$\begin{aligned} P(A = \text{true}) &= a \\ P(B = \text{true}) &= b \\ P(C = \text{true}) &= c. \end{aligned}$$

For *and*-combinations, Q is true only when A, B, and C are all true and we have a matrix of the form

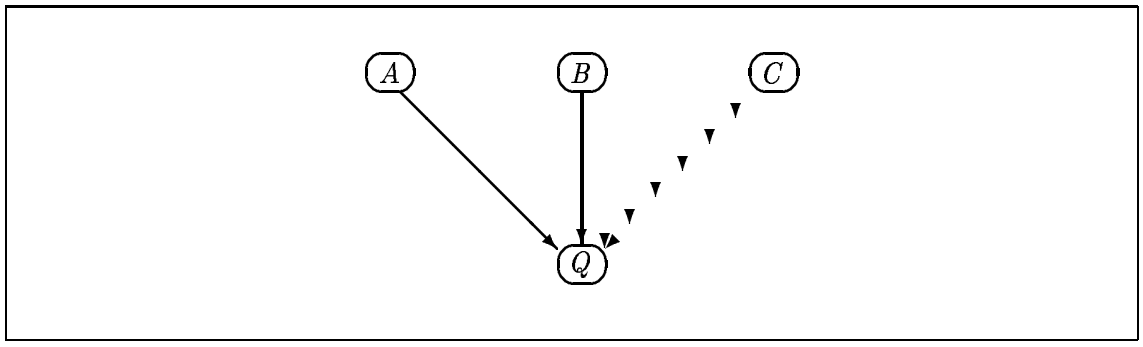

Figure 4.5: Network for link matrix examples

$$L_{\text{and}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using a closed form update procedure we have

$$\begin{aligned} P(Q = \mathit{true}) &= abc && (4.1) \\ P(Q = \mathit{false}) &= (1-a)(1-b)(1-c) + (1-a)(1-b)c && (4.2) \\ &\quad +(1-a)b(1-c) + (1-a)bc + a(1-b)(1-c) \\ &\quad +a(1-b)c + ab(1-c) \\ &= 1 - abc && (4.3) \end{aligned}$$

which is the familiar rule for conjunctive combination of events.

For *or*-combinations, Q will be true when any of A, B, or C is true and false only when A, B, and C are all false. This gives a link matrix of the form

$$L_{\text{or}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Again using a closed form of the update procedures, we have

$$\begin{aligned} P(Q = \mathit{true}) &= (1-a)(1-b)c + (1-a)b(1-c) + (1-a)bc && (4.4) \\ &\quad +a(1-b)(1-c) + a(1-b)c + ab(1-c) + abc \end{aligned}$$

$$
\begin{aligned}
&= a + b + c - (ab + bc + ac) + abc \\
&= 1 - (1-a)(1-b)(1-c) && (4.5) \\
P(Q = \mathit{false}) &= (1-a)(1-b)(1-c) && (4.6) \\
& && (4.7)
\end{aligned}
$$

which is the familiar rule for disjunctive combination of events that are not known to be mutually exclusive.

The *not* operator is defined only for unary propositions or nodes with a single parent. If Q has the single parent A, $Q = \mathit{true}$ exactly when $A = \mathit{false}$ which gives a link matrix of the form

$$L_{\text{not}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and results in

$$
\begin{aligned}
P(Q = \mathit{true}) &= 1 - a && (4.8) \\
P(Q = \mathit{false}) &= a. && (4.9)
\end{aligned}
$$

If we restrict the parent nodes for any of the logic operators to values 0 or 1 then Q must also have a value of 0 or 1. If we allow terms to take on weights in the range $[0, 1]$ and interpret these weights as the probability that the term has been assigned to a document text, then these inference networks provide a natural interpretation for Boolean retrieval with weighted indexing. The use of these canonical forms to simulate Boolean retrieval will be discussed in section 6.3.

A fourth link matrix form arises when our belief in Q depends only on the number of parents that are true. If j corresponds to a link matrix column number for which m parents are true (in which m bits $= 1$), then

$$L_{\text{sum}}[1, j] = \frac{m}{n}$$

$$L_{\text{sum}}[0,j] = \frac{n-m}{n}.$$

Thus, for our three parent example

$$L_{\text{sum}} = \begin{pmatrix} 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & 1 \end{pmatrix}.$$

Evaluation of this *sum* link matrix results in

$$\begin{aligned} P(Q = \mathit{true}) &= \frac{1}{3}(1-a)(1-b)c + \frac{1}{3}(1-a)b(1-c) + \frac{2}{3}(1-a)bc \\ &\quad + \frac{1}{3}a(1-b)(1-c) + \frac{2}{3}a(1-b)c + \frac{2}{3}ab(1-c) + abc \\ &= \frac{a+b+c}{3} \\ P(Q = \mathit{false}) &= (1-a)(1-b)(1-c) + \frac{2}{3}(1-a)(1-b)c + \frac{2}{3}(1-a)b(1-c) \\ &\quad + \frac{1}{3}(1-a)bc + \frac{2}{3}a(1-b)(1-c) + \frac{1}{3}a(1-b)c + \frac{1}{3}ab(1-c) \\ &= 1 - \frac{a+b+c}{3} \end{aligned}$$

In this matrix form all parents are weighted equally; if all parents are observed to be true then $P(Q = \mathit{true})$ is three times greater than if one parent is observed. A number of other weightings are possible. For example, we can choose weights so that Q is true when any m parents are true to implement an "m-of-n" operator [Ang75], or we can choose weights so that the first parent observed has the most influence on our belief in Q and the second and third parents have less influence on our belief (essentially, the *or*-combination is an extreme case in which only the first parent influences our belief). Similarly, we can choose weights so that the first parent observed has little or no influence on our belief in Q and the second and third parents determine our belief.

The final link matrix form we will present here is a generalization of the *sum* matrix in which each parent has a weight associated with it, as does the child. In

this weighted-sum (*wtd_sum*) matrix, our belief in Q depends on the specific parents that are true – parents with larger weights have more influence in our belief. The weight at Q acts to set the maximum belief that can be achieved at Q. If we let w_a, w_b, and w_c be the parent weights, $0 \leq w_q \leq 1$ the child weight, and $t = w_a + w_b + w_c$ for our example, then we have a link matrix of the form

$$\begin{pmatrix} 1 & 1-\frac{w_c w_q}{t} & 1-\frac{w_b w_q}{t} & 1-\frac{(w_b+w_c)w_q}{t} & 1-\frac{w_a w_q}{t} & 1-\frac{(w_a+w_c)w_q}{t} & 1-\frac{(w_a+w_b)w_q}{t} & 1-w_q \\ 0 & \frac{w_c w_q}{t} & \frac{w_b w_q}{t} & \frac{(w_b+w_c)w_q}{t} & \frac{w_a w_q}{t} & \frac{(w_a+w_c)w_q}{t} & \frac{(w_a+w_b)w_q}{t} & w_q \end{pmatrix}.$$

Evaluation of this link matrix form results in

$$\begin{aligned} P(Q = true) &= \frac{w_c w_q}{t}(1-a)(1-b)c + \frac{w_b w_q}{t}(1-a)b(1-c) \\ &\quad + \frac{(w_b+w_c)w_q}{t}(1-a)bc + \frac{w_a w_q}{t}a(1-b)(1-c) \\ &\quad + \frac{(w_a+w_c)w_q}{t}a(1-b)c + \frac{(w_a+w_b)w_q}{t}ab(1-c) + w_q abc \\ &= \frac{(w_a a + w_b b + w_c c)w_q}{t} \qquad (4.10) \\ P(Q = false) &= w_q(1-a)(1-b)(1-c) + \frac{(w_a+w_b)w_q}{t}(1-a)(1-b)c \\ &\quad + \frac{(w_a+w_c)w_q}{t}(1-a)b(1-c) + \frac{w_a w_q}{t}(1-a)bc \\ &\quad + \frac{(w_b+w_c)w_q}{t}a(1-b)(1-c) + \frac{w_b w_q}{t}a(1-b)c \\ &\quad + \frac{w_c w_q}{t}ab(1-c) \\ &= 1 - \frac{(w_a a + w_b b + w_c c)w_q}{t} \end{aligned}$$

The *sum* matrix is a special case of *wtd_sum* where all weights are 1.

The *wtd_sum* link matrix can be used to implement a variety of weighting schemes, including the familiar term weighting schemes based on within-document term frequency (*tf*), inverse document frequency (*idf*) or both (*tf.idf*). To illustrate a *tf.idf* weighting, let Q in our example be a representation node and let A, B, and

C be document nodes. Let w_a, w_b, and w_c be the normalized tf values for A, B, and C and let

$$w_q = idf_q(w_a + w_b + w_c) \tag{4.11}$$

Given our basic model, when A is instantiated, belief in Q is given by

$$\begin{aligned} \text{bel}(Q) &= \frac{w_a w_q}{w_a + w_b + w_c} \\ &= \frac{tf_a \cdot idf_q(w_a + w_b + w_c)}{w_a + w_b + w_c} \\ &= tf_a \cdot idf_q \end{aligned}$$

which is a common form of $tf.idf$ weight. Similarly, when B is instantiated, $\text{bel}(Q) = tf_b \cdot idf_q$. In general, when a document is instantiated all representation concept nodes to which it is attached take of the $tf.idf$ weight associated with the document/term pair.

The weight at Q has two distinct parts. The first part (idf_q in our example) acts to set the maximum belief achievable at a node. If, for some combination of parent values, our belief in Q is certain then this component disappears. Note that in this formulation, the idf component is dependent only upon the distribution of the term in the collection, not on the distribution of the term in relevant and non-relevant subsets. Relevance feedback is modeled as part of the query network and does not affect belief in representation concepts.

The second part ($w_a + w_b + w_c$ in our example) acts to normalize the parent weights. Equation 4.11 is the appropriate weight for the basic model in which only one document can be instantiated at a time. In the extended model of Chapter 7 where multiple documents can be instantiated, this component is adjusted to normalize for the maximum achievable set of parent weights. In the general case,

where all parents can take any value in the range $[0, 1]$, this normalizing component disappears.

While these five canonical forms are sufficient for the retrieval inference networks described here, many others are possible (see section 6.3.2). Further, when n is small (say, less than 5 or 6) we can use the full link matrix if the dependence of a node on its parents does not fit a canonical form.

To summarize the results of this section, the following closed-form expressions can be used to evaluate the canonical matrices for a node Q with parents $P_1, \ldots, P_n$ where $P(P_1 = true) = p_1, \ldots, P(P_n = true) = p_n$:

$$\text{bel}_{\text{or}}(Q) = 1 - (1 - p_1) \cdot \ldots \cdot (1 - p_n) \tag{4.12}$$

$$\text{bel}_{\text{and}}(Q) = p_1 \cdot p_2 \cdot \ldots \cdot p_n \tag{4.13}$$

$$\text{bel}_{\text{not}}(Q) = 1 - p_1 \tag{4.14}$$

$$\text{bel}_{\text{sum}}(Q) = \frac{p_1 + p_2 + \cdots + p_n}{n} \tag{4.15}$$

$$\text{bel}_{\text{wtd\_sum}}(Q) = \frac{(w_1 p_1 + w_2 p_2 + \cdots + w_n p_n) w_q}{w_1 + w_2 + \cdots + w_n} \tag{4.16}$$

In the network model, the distinction between the Boolean operators and the probabilistic sum operators begins to blur. The operators are, after all, only specialized link matrix forms and the model allows Boolean and probabilistic operators to be freely mixed in expressions. The ability to mix operator types is required to allow us to combine query forms and is useful in representing phrases and for developing relevance feedback strategies (discussed further in Chapter 7).

4.5.2 Time and space complexity

The canonical link matrix forms of the last section are intended to mitigate the exponential space and time complexity associated with a full link matrix. We next consider the space and time complexity of the canonical forms.

Table 4.1: Complexity of canonical forms for a node with n parents

matrix form	space complexity	time complexity
not	$O(1)$	$O(1)$
and	$O(1)$	$O(n)$
or	$O(1)$	$O(n)$
sum	$O(1)$	$O(n)$
wtd_sum	$O(n)$	$O(n)$

Table 4.1 summarizes the space complexity of each canonical matrix form. The *sum* and logic operators do not require any link matrix information and therefore require $O(1)$ space. The *wtd_sum* operator requires a weight for each parent and therefore uses $O(n)$ space. Clearly, these canonical forms substantially reduce space requirements.

For time complexity, we will take the number of factors that must be multiplied to compute a term in an expression as the basic cost associated with evaluating that term. Without loss of generality, we also assume that the cost of evaluating a closed form expression is the sum of the costs of evaluating the terms in the expression.

A lower bound for all of the canonical forms is $\Omega(n)$ since each parent belief must be accessed in order to compute belief. Any algorithm that does not use all parent beliefs cannot be guaranteed to produce correct results.

Given our assumption that *not* is defined only for single parent nodes, bel_{not} (equation 4.14) requires a single access to its only parent and operates in $O(1)$ time.

From equation 4.13, the expression for bel_{and} contains a single term which requires access to each parent so bel_{and} can be computed in $O(n)$ time. Similarly, from equation 4.12, the expression for bel_{or} contains two terms, one is a constant and the other requires access to each parent so bel_{or} can also be computed in $O(n)$ time.

The closed-form expressions for *sum* (equation 4.15) and *wtd_sum* (equation 4.16) contain n terms, each requiring access to a single parent, so *sum* and *wtd_sum* can both be computed in $O(n)$ time.

4.5.3 Diagnostic evidence

The canonical link matrix forms of section 4.5.1 provide closed-form expressions for use in propagating predictive evidence (from parents to children). Except for evidence attached to documents (roots), all evidential flow in retrieval inference networks is predictive. Finding useful canonical forms for diagnostic evidence (children to parents) is less straightforward.

When combining evidence from parents, the entire link matrix and all parent messages are used to compute the predictive component of a node's belief. When computing the diagnostic message to be sent to each parent P_i the effect of evidence received from P_i is excluded. In order to exclude P_i we essentially use the original link matrix which specifies $P(Q|P_1, \ldots, P_n)$ to form a new link matrix which specifies $P(Q|P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n)$. Conceptually, this link matrix is multiplied by the outer product of all messages from parents other than P_i and this result is combined with the diagnostic evidence provided by Q's children to form the diagnostic message for P_i. For diagnostic evidence, then, we need n different link matrices, each of which is derived by "summing out" the effect of one parent.

In order for canonical forms to be useful for diagnostic evidence, we would like matrices that produce the same result (or at least a small number of different results) when any parent is summed out. The matrix forms for *and*, *or*, and *not* have this property, but most canonical forms, including *sum* and *wtd_sum*, do not. Returning to the example of Figure 4.5, consider a link matrix of the form

$$L = \begin{pmatrix} 1-l_0 & 1-l_1 & 1-l_2 & 1-l_3 & 1-l_4 & 1-l_5 & 1-l_6 & 1-l_7 \\ l_0 & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 \end{pmatrix}.$$

For clarity, we will show only the row for $q = 1$ in the remainder of this example. When summing out A, we get a new matrix of the form

$$L_{BC} = \begin{pmatrix} l_0 + l_4 & l_1 + l_5 & l_2 + l_6 & l_3 + l_7 \end{pmatrix}.$$

Similarly, when summing out B and C we get

$$L_{AC} = \begin{pmatrix} l_0 + l_2 & l_1 + l_3 & l_4 + l_6 & l_5 + l_7 \end{pmatrix}.$$
$$L_{AB} = \begin{pmatrix} l_0 + l_1 & l_2 + l_3 & l_4 + l_5 & l_6 + l_7 \end{pmatrix}.$$

In order to have a single reduced matrix for this example, then, we need an original matrix which satisfies the following constraints

$$l_0 + l_4 = l_0 + l_2 = l_0 + l_1 \tag{4.17}$$
$$l_1 + l_5 = l_1 + l_3 = l_2 + l_3$$
$$l_3 + l_7 = l_5 + l_7 = l_6 + l_7. \tag{4.18}$$

A matrix satisfying these constraints will provide the same result when any parent is summed out. Moreover, to be generally useful, the matrix form must satisfy a similar set of constraints for any number of parents. Clearly, the *sum* and *wtd_sum* forms do not satisfy these constraints.

Since *not* is defined only for one parent, the canonical form is trivial. For *and*, l_i is 0 for $i < 2^n - 1$ and $l_{2^n-1} = 1$, so every sum in the above set of constraints that does not involve l_{2^n-1} is 0 and every sum involving l_{2^n-1} is 1. Since every sum involving l_{2^n-1} appears in the last constraint above (4.18), *and* satisfies the constraints for $n = 3$. A similar argument can be used to show that *and* satisfies the constraints for arbitrary n.

For *or*, $l_0 = 0$ and $l_i = 1, 0 < i \leq 2^n - 1$, so every sum in the above constraints that involves l_0 is 1 and every sum that does not involve l_0 is 2. All sums involving l_0 occur in the first constraint above (4.17) so *or* satisfies the constraints for $n = 3$. Again, a similar argument can be used to show that *or* satisfies the constraints for arbitrary n.

In practice, we do not form these reduced link matrices directly but compute them incrementally by forming the outer product of the link matrix and the message from a single parent and then contracting on the common index. The contracted matrix is then used to form the outer product with the next parent, and so on (see Appendix). For diagnostic evidence we will form this product and contract $n - 1$ times which results in a matrix (an order 2 tensor) which specifies $P(e^+|P_i)$ where e^+ is the combined evidence from all parents excluding P_i. This matrix has a canonical form for the logic operators, but not in general. If we let p be the result of the closed-form expressions for belief (equations 4.12 and 4.13), then this matrix has the form

$$\begin{pmatrix} 1 & 1-p \\ 0 & p \end{pmatrix}$$

for *and*, and

$$\begin{pmatrix} 1-p & 0 \\ p & 1 \end{pmatrix}$$

for *or*.

Our use of diagnostic evidence in retrieval networks is limited, in part because of the difficulty of finding exact closed-form solutions for combining functions other than the Boolean operators. The fact that canonical forms are rare suggests that any use of diagnostic evidence for nodes with a large number of parents will require some form of approximation that allows efficient computation.

Chapter 5

Inference network example

In this chapter we will present an example inference network and show how queries are evaluated.

The inference network fragment shown in figure 5.1 contains two documents and four representation concepts. Document d_1 discusses the use of inference networks for information retrieval and is represented by the phrase *inference network* and the keywords *information* and *retrieval* (among others). Document d_2 discusses the retrieval of satellites from low-earth orbit and is represented by the keywords *information*, *retrieval*, and *satellite*. A single query has been attached containing the phrase *inference network* and the keywords *information* and *retrieval*. For the purposes of this example we are using only features of the simplified form of the basic model shown in Figure 4.2. We will use this network to estimate bel$(Q|d_1)$ and bel$(Q|d_2)$.

In a belief network the absence of any evidential support for or against a proposition is represented by bel $= 0.5$. Positive evidential support is represented by beliefs in the range $(0.5, 1.0]$ with bel $= 1.0$ representing certainty that the proposition is true. Similarly, negative evidential support is represented by beliefs in the range $[0.0, 0.5)$ with bel $= 0.0$ representing certainty that the proposition is

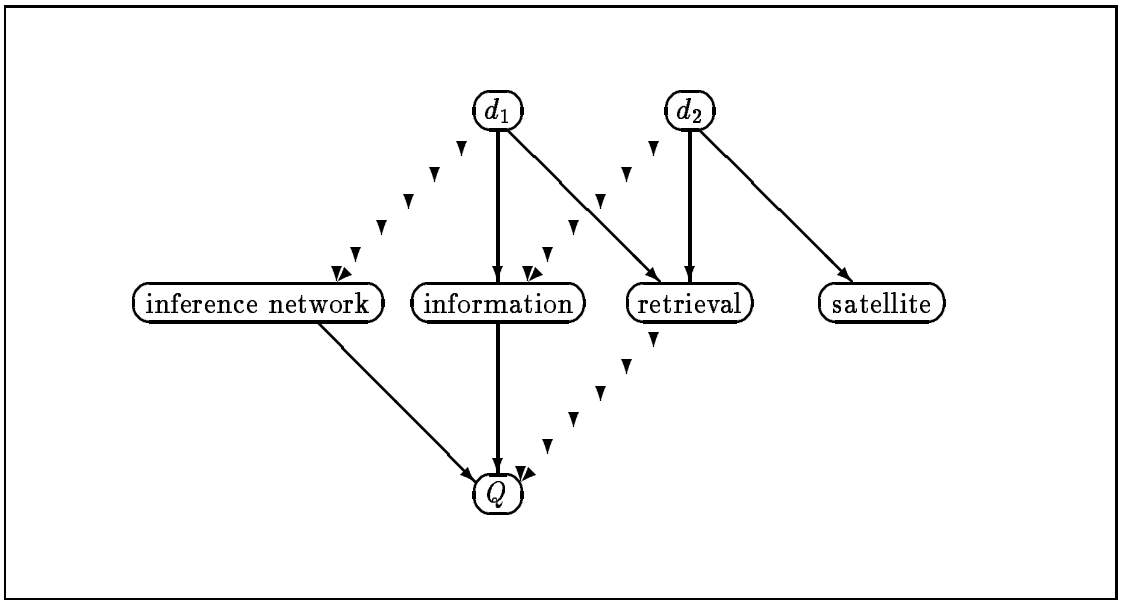

Figure 5.1: Inference network fragment

false. Our first task, then, is to find estimates for belief that lie in the appropriate intervals.

Several weighting schemes have been proposed in which the belief in a representation concept depends on the frequency with which the concept occurs in a document and on the frequency of the concept in the collection (see, for example, [EW61]). We will assume that belief in a representation concept is proportional to the within-document frequency (tf) and inversely proportional to the frequency of the concept in the collection. The collection frequency component is generally expressed as the term's inverse document frequency (idf) which is given by

$$idf = \log(\frac{\text{collection size}}{\text{concept frequency}}).$$

We will normalize both tf and idf to the range $[0, 1]$ by dividing tf by the maximum tf value for any term in the document and dividing idf by the maximum possible

idf value in the collection (the *idf* score for a term that occurs once). For concept i that occurs tf_{ij} times in document j and f_i times in the entire collection, we have

$$ntf_{ij} = \frac{tf_{ij}}{max\_tf_j} \tag{5.1}$$

$$nidf_i = \frac{\log(\frac{\text{collection size}}{f_i})}{\log(\text{ collection size})}. \tag{5.2}$$

Techniques for estimating these beliefs are discussed in detail in Chapter 8, but for the purposes of the example, we will assume that $P(r_i = \text{true}|d_j = \text{true})$ is given by

$$P(r_i = \text{true}|d_j = \text{true}) = 0.5 + (0.5 \cdot ntf_{ij} \cdot nidf_i) \tag{5.3}$$

and that

$$P(r_i = \text{true}|\text{all parents false}) = 0.0. \tag{5.4}$$

(As will be discussed in Chapter 8, equation 5.4 is not a very good estimate, but it simplifies the example and is the estimate used in most probabilistic models.) Link matrices can be built directly from these estimates.

Arcs are drawn from a document only to representation concepts that have been assigned to that document. When a document is instantiated it provides equal support for all members of the set of assigned representation concepts; all other representation concepts receive no support (this is not the case for the extended model of Chapter 7). Any representation concept with no support is believed to be false (not observed or bel = 0). Any representation concept that receives support is believed to the degree specified in equation 5.3.

Table 5.1 gives frequency and *nidf* scores based on a small NTIS database (n = 136,609) and the *tf* and *ntf* values for the two documents. We assume that $max\_tf_{d_1} = 5$ and that $max\_tf_{d_2} = 4$.

Table 5.1: Frequencies and *idf* and *tf* weights

	frequency	*nidf* score	tf_{d_1}	tf_{d_2}	ntf_{d_1}	ntf_{d_2}
inference network	16	0.77	3	0	0.6	0.0
information	16461	0.18	3	2	0.6	0.5
retrieval	820	0.43	5	1	1.0	0.25
satellite	2675	0.33	0	4	0.0	1.0

5.1 Probabilistic query

If we interpret the query in figure 5.1 as a probabilistic natural language query, from equation (5.3) we have

$$\begin{aligned} P(\textit{inference network} = \textit{true} | d_1 = \textit{true}) &= 0.5 + 0.5 \cdot 0.6 \cdot 0.77 \\ &= 0.731 \end{aligned}$$

which results in a link matrix of

$$L_{\textit{inference net}} = \begin{pmatrix} 1.000 & 0.269 \\ 0.000 & 0.731 \end{pmatrix}.$$

For the *information* node we must compute beliefs for both parents, so

$$\begin{aligned} P(\textit{information} = \textit{true} | d_1 = \textit{true}) &= 0.5 + 0.5 \cdot 0.6 \cdot 0.18 \\ &= 0.554 \\ P(\textit{information} = \textit{true} | d_2 = \textit{true}) &= 0.5 + 0.5 \cdot 0.5 \cdot 0.18 \\ &= 0.545 \end{aligned}$$

which results in a link matrix of

$$L_{\textit{information}} = \begin{pmatrix} 1.000 & 0.455 & 0.446 & 0.446 \\ 0.000 & 0.545 & 0.554 & 0.554 \end{pmatrix}.$$

The last column of this link matrix is unused since only one document can be instantiated at a time. It is set to the maximum of the individual document beliefs.

Using the same procedure, the link matrix for *retrieval* is

$$L_{retrieval} = \begin{pmatrix} 1.000 & 0.285 & 0.446 & 0.285 \\ 0.000 & 0.715 & 0.554 & 0.715 \end{pmatrix}$$

and for *satellite* we have

$$L_{satellite} = \begin{pmatrix} 1.000 & 0.335 \\ 0.000 & 0.665 \end{pmatrix}.$$

There are several ways to estimate the matrix at Q. We would generally estimate the matrix based on the frequency of each term in the query text, but for the example we will assume that the user has indicated that the probability that a document matches his information need if it contains none of the query terms is 0.1, that the probability for a document containing all of the terms is 0.9, that the phrase *inference network* is twice as important as either keyword, and that the probabilities for multiple terms are additive. The link matrix can then be estimated as

$$L_Q = \begin{pmatrix} 0.9 & 0.7 & 0.7 & 0.5 & 0.5 & 0.3 & 0.3 & 0.1 \\ 0.1 & 0.3 & 0.3 & 0.5 & 0.5 & 0.7 & 0.7 & 0.9 \end{pmatrix}.$$

Instantiating d_1 results in

$$\begin{array}{rclrcl} \text{bel(inference network)} & = & 0.731 & \text{bel(information)} & = & 0.554 \\ \text{bel(retrieval)} & = & 0.554 & \text{bel(satellite)} & = & 0.000 \end{array}$$

which gives

$$\begin{aligned} \text{bel}(Q|d_1) = \; & 0.1 \cdot 0.269 \cdot 0.446 \cdot 0.446 + 0.3 \cdot 0.269 \cdot 0.446 \cdot 0.554 \\ & +0.3 \cdot 0.269 \cdot 0.554 \cdot 0.446 + 0.5 \cdot 0.269 \cdot 0.554 \cdot 0.554 \\ & +0.5 \cdot 0.731 \cdot 0.446 \cdot 0.446 + 0.7 \cdot 0.731 \cdot 0.446 \cdot 0.554 \\ & +0.7 \cdot 0.731 \cdot 0.554 \cdot 0.446 + 0.9 \cdot 0.731 \cdot 0.554 \cdot 0.554 \end{aligned}$$

$$= \quad 0.614.$$

Instantiating d_2 results in

$$\begin{array}{rcl@{\qquad}rcl} \text{bel(inference network)} & = & 0.000 & \text{bel(information)} & = & 0.545 \\ \text{bel(retrieval)} & = & 0.715 & \text{bel(satellite)} & = & 0.665 \end{array}$$

which gives

$$\begin{aligned} \text{bel}(Q|d_2) &= 0.1 \cdot 1 \cdot 0.455 \cdot 0.285 + 0.3 \cdot 1 \cdot 0.455 \cdot 0.715 + 0.3 \cdot 1 \cdot 0.545 \cdot 0.285 \\ &\quad +0.5 \cdot 1 \cdot 0.545 \cdot 0.715 + 0.5 \cdot 0 \cdot 0.455 \cdot 0.285 + 0.7 \cdot 0 \cdot 0.455 \cdot 0.715 \\ &\quad +0.7 \cdot 0 \cdot 0.545 \cdot 0.285 + 0.9 \cdot 0 \cdot 0.545 \cdot 0.715 \\ &= 0.352. \end{aligned}$$

If relevance judgements were available, they could be used to adjust link matrix values at Q and to produce refined estimates of bel(Q).

5.2 Boolean query – weighted indexing

If the query of figure 5.1 is interpreted as the Boolean conjunction

"inference net" *and* information *and* retrieval

rather than a natural language query, we would use the link matrix form described in section 4.5.1 at Q and

$$L_Q = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using the same term weights as above our beliefs in the representation concepts would be unchanged and evaluation of the Boolean query would result in

$$\begin{aligned} \text{bel}(Q|d_1) &= 0 \cdot 0.269 \cdot 0.446 \cdot 0.446 + 0 \cdot 0.269 \cdot 0.446 \cdot 0.554 \\ &\quad +0 \cdot 0.269 \cdot 0.554 \cdot 0.446 + 0 \cdot 0.269 \cdot 0.554 \cdot 0.554 \end{aligned}$$

$$
\begin{aligned}
& +0 \cdot 0.731 \cdot 0.446 \cdot 0.446 + 0 \cdot 0.731 \cdot 0.446 \cdot 0.554 \\
& +0 \cdot 0.731 \cdot 0.554 \cdot 0.446 + 1 \cdot 0.731 \cdot 0.554 \cdot 0.554 \\
= \; & 0.224.
\end{aligned}
$$

and

$$
\begin{aligned}
\text{bel}(Q|d_2) = \; & 0 \cdot 1 \cdot 0.455 \cdot 0.285 + 0 \cdot 1 \cdot 0.455 \cdot 0.715 + 0 \cdot 1 \cdot 0.545 \cdot 0.285 \\
& +0 \cdot 1 \cdot 0.545 \cdot 0.715 + 0 \cdot 0 \cdot 0.455 \cdot 0.285 + 0 \cdot 0 \cdot 0.455 \cdot 0.715 \\
& +0 \cdot 0 \cdot 0.545 \cdot 0.285 + 1 \cdot 0 \cdot 0.545 \cdot 0.715 \\
= \; & 0.0.
\end{aligned}
$$

5.3 Conventional Boolean query

If we interpret the query of figure 5.1 as a conventional Boolean conjunction with binary rather that weighted indexing, instantiating d_1 results in

$$
\begin{array}{rclrcl}
\text{bel(inference network)} & = & 1.0 & \text{bel(information)} & = & 1.0 \\
\text{bel(retrieval)} & = & 1.0 & \text{bel(satellite)} & = & 0.0
\end{array}
$$

and instantiating d_2 results in

$$
\begin{array}{rclrcl}
\text{bel(inference network)} & = & 0.0 & \text{bel(information)} & = & 1.0 \\
\text{bel(retrieval)} & = & 1.0 & \text{bel(satellite)} & = & 1.0
\end{array}
.
$$

Using the link matrix for *and* gives

$$
\begin{aligned}
\text{bel}(Q|d_1) = \; & 0 \cdot 0 \cdot 0 \cdot 0 + 0 \cdot 0 \cdot 0 \cdot 1 + 0 \cdot 0 \cdot 1 \cdot 0 \\
& +0 \cdot 0 \cdot 1 \cdot 1 + 0 \cdot 1 \cdot 0 \cdot 0 + 0 \cdot 1 \cdot 0 \cdot 1 \\
& +0 \cdot 1 \cdot 1 \cdot 0 + 1 \cdot 1 \cdot 1 \cdot 1 \\
= \; & 1.0
\end{aligned}
$$

and

$$
\text{bel}(Q|d_2) = 0 \cdot 1 \cdot 0 \cdot 0 + 0 \cdot 1 \cdot 0 \cdot 1 + 0 \cdot 1 \cdot 1 \cdot 0
$$

$$
\begin{aligned}
& +0 \cdot 1 \cdot 1 \cdot 1 + 0 \cdot 0 \cdot 0 \cdot 0 + 0 \cdot 0 \cdot 0 \cdot 1 \\
& +0 \cdot 0 \cdot 1 \cdot 0 + 1 \cdot 0 \cdot 1 \cdot 1 \\
= \; & 0.0
\end{aligned}
$$

which is equivalent to conventional Boolean evaluation.

These examples illustrate the use of the inference networks and have been simplified to reduce the number of computational details. We have used simple estimates for the link matrices, and have not dealt with the more complex network features described in Chapter 7.

Chapter 6

Comparison with other retrieval models

The inference network retrieval model generalizes both the probabilistic and Boolean models. Inference networks can be used to simulate both probabilistic and Boolean queries and can be used to combine results from multiple queries.

In this chapter we will compare the inference network model with probabilistic (Sections 6.1 and 6.2) and Boolean (Section 6.3) models and show how inference networks can be used to simulate both forms of retrieval. We then consider how the probabilities required by the model can be estimated (Section 6.4) and show that the estimation problems are essentially equivalent to those encountered with probabilistic or vector-space retrieval.

6.1 Probabilistic retrieval models

Conventional probabilistic models [van79, BC87, SM83] rank documents by the probability that each document in the collection would be judged relevant to a given query, $P(\text{relevant}|d_i)$.[1] This is, in many ways, similar to computing the probability

[1]Most probabilistic models do not actually compute $P(\text{relevant}|d_i)$, but simply rank documents using some function that is monotonic with $P(\text{relevant}|d_i)$. Like Fuhr ([Fuh89b]), we believe that

that a user's information need is met given a specific document, $P(I|d_i)$. The principal differences between conventional probabilistic models and the model described here are: 1) conventional probabilistic models do not explicitly represent the query, 2) conventional probabilistic models do not distinguish between a document and its representations but treat a document as a single vector, and 3) the inference model depends less upon Bayesian inversion than probabilistic models, Bayesian inversion is just one way to estimate $P(I|d_i)$ (or $P(Q|d_i)$ in the case of a single query).

In this section we first summarize the major differences between the inference network and conventional probabilistic models by comparing the network model to the binary independence model. We then provide a formal comparison of the inference network model with a recent probabilistic model that explicitly represents documents and queries.

6.1.1 Binary Independence Model

An inference network that corresponds to the binary independence model [van79, Fuh89a] is shown in figure 6.1. A document is represented by a vector whose components are indexing or representation concepts ($d_i = \{r_1, \ldots, r_n\}$). The set of concepts considered is generally restricted to the subset that actually occurs in the query. Comparing this network with that shown in figure 4.1, we see that in the binary independence model, the document network is represented by a single level of representation nodes and the query network consists of a single relevance node. In order to implement this network we must somehow estimate the probability of relevance given the set of parent representation concepts and this estimate must

an estimate of the probability of relevance is more useful than the ranking by itself. A ranked list of documents in which the top ranked document has a probability of relevance of 0.5 should be viewed differently than a similar list in which the top ranked document has a probability of relevance of 0.95.

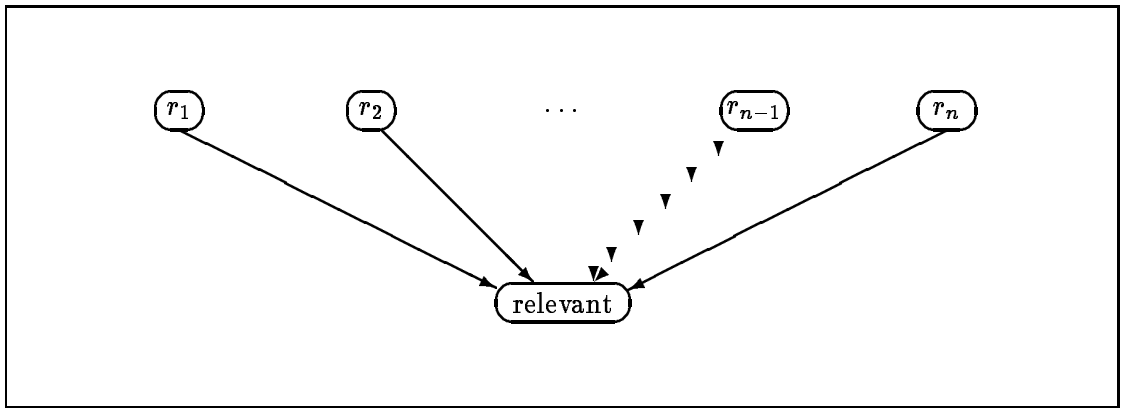

Figure 6.1: Inference network for binary independence model

incorporate all of our judgments about the probability that a representation concept should be assigned to a document, about the semantic and stochastic relationships between representation concepts, about the relationship between concepts named in the query and assigned to documents, and about the semantics of the query itself. This dependence is complex and its estimation is not a task we could expect users to perform willingly or reliably.

One approach to simplifying the estimation task is to invoke Bayes' rule so that we need only estimate the probability that each representation concept occurs in relevant or non-relevant documents. This approach does not help to provide initial estimates of the probability distributions since these "simpler" estimates must still incorporate all of the judgments required for the "hard" estimate. The advantage of this approach is that, given samples of relevant and non-relevant documents, it is easy to compute $P(r_i)$ for the relevant sample and to use the result as an estimate of $P(r_i|\text{relevant} = \textit{true})$ and similarly for $P(r_i|\text{relevant} = \textit{false})$. Given a set of independence assumptions and estimates for $P(d_i)$ and $P(\text{relevant} = \textit{true})$ we can

compute $P(\text{relevant}|d_i)$.[2] Estimating $P(\text{relevant}|d_i)$ without the use of Bayes' rule would be extremely difficult [LCB89].

Essentially the same procedures can be used to estimate $P(Q|d_i)$. If we assume a one-to-one correspondence between representation and query concepts then the estimation procedures are equivalent. If we do not make this assumption then we must compute $P(c_j|\pi_{c_j})$ and find an expected value for $P(c_j|d_i)$ in order to estimate $P(Q|d_i)$.

The question remains, however, whether estimates of $P(\text{relevant}|d_i)$ or $P(Q|d_i)$ obtained in this way match users' intuition about the dependence. The fact that relevance feedback does improve retrieval performance suggests that the estimates of $P(\text{relevant}|d_i)$ do capture at least some of the dependence, but these estimates are generally based on a small number of relevant documents and are necessarily rather coarse.

While it is clear that estimating $P(\text{relevant}|d_i)$ directly from a small number of documents is impractical, it may be possible to obtain estimates of $P(Q|\pi_Q)$. Users may, for example, be able to assign importance to the concepts in their query and may be able to identify significant interactions between concepts. These estimates could improve the initial estimate and might be used in conjunction with the estimates derived from training samples.

A second approach to simplifying the estimation task is to identify the different types of judgments that enter into the overall estimate and to develop estimates for each type of judgment separately. The model presented here represents one decomposition in which the task of estimating the probability that a given document satisfies an information need consists of judgments about the relationship

[2] $P(d_i)$ and $P(\text{relevant} = \textit{true})$ do not play a major role in probabilistic models that only produce a document ranking but are required to compute $P(\text{relevant}|d_i)$.

of a document to its text, the assignment of representation concepts to the text, the relationships between query and representation concepts, and the relationship between queries, query concepts, and the information need. Other decompositions are certainly possible and can be accommodated within the same general framework. The set of relationships presented here incorporates those judgments most important for current generation document retrieval systems.

When viewed this way, the probabilistic and inference models use two similar approaches to the same estimation problem. The probabilistic model uses a single, general purpose rule and makes assumptions about term dependence in order to estimate $P(\text{relevant}|d_i)$. The model presented here views the problem of estimating $P(I|d_i)$ as consisting of a set of logically related estimates. Each estimate is made independently using procedures specific to the type of estimate; the "probabilistic" estimate of $P(Q|\pi_Q)$ is simply one component of the overall estimate. The component estimates are then combined in a manner consistent with the dependence relationships represented in the inference network to provide an estimate of $P(I|d_i)$.

6.1.2 Comparison with the RPI model

To further clarify the relationship between the inference network model and the probabilistic model, we will compare the inference network model with Fuhr's model for retrieval with probabilistic indexing (RPI model) [Fuh89a]. To simplify the comparison, we will temporarily adopt Fuhr's notation. Let

d_m represent a document in the collection,

$\mathbf{x}$ be the binary vector $(x_1, x_2, \ldots, x_n)$ in which each x_i corresponds to a document descriptor r_i (a representation concept),

f_k represent the query, and

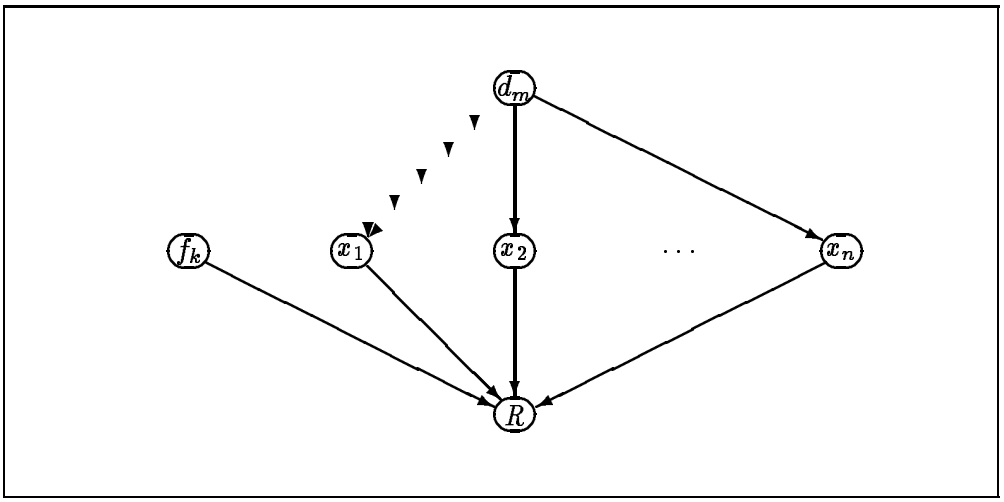

Figure 6.2: Inference network for the RPI model

R represent the event that a document is judged relevant to a query.

All variables are binary-valued. In this model, $P(x_i = 1|d_m)$ is interpreted as the probability that a descriptor r_i is a "correct" indexing of d_m. Let X be the set of possible values for $\mathbf{x}$, where $|X| \leq 2^n$.

The network shown in figure 6.2 corresponds to the probability distribution

$$\begin{aligned} P(R, f_k, x_1, \ldots, x_n, d_m) &= P(R|f_k, d_m) \\ &= P(R|f_k, x_1, \ldots, x_n)P(x_1|d_m)\ldots P(x_n|d_m)P(f_k)P(d_m). \end{aligned}$$

We will evaluate this expression for a given document and query so f_k and d_m are known and the distribution reduces to

$$P(R|f_k, d_m) = P(R|f_k, x_1, \ldots, x_n)P(x_1|d_m)\ldots P(x_n|d_m).$$

Assuming that the descriptors are assigned independently, that is

$$P(\mathbf{x}|d_m) = \prod_{1 \leq i \leq n} P(x_i|d_m),$$

the basic ranking expression for the network of figure 6.2 is

$$P(R|f_k, d_m) = \sum_{\mathbf{x} \in X} P(R|f_k, \mathbf{x}) P(\mathbf{x}|d_m). \tag{6.1}$$

Equation 6.1 is equivalent to the basic ranking expression used by Fuhr [Fuh89a, equation 9]. Equation 6.1 can be expanded to the familiar product form

$$P(R|f_k, d_m) = P(R|f_k) \prod_{1 \le i \le n} \left(\frac{p_{ik}}{q_i} u_{im} + \frac{1 - p_{ik}}{1 - q_i} (1 - u_{im}) \right) \tag{6.2}$$

where

$$\begin{aligned} p_{ik} &= P(x_i = 1|R, f_k) \\ q_i &= P(x_i = 1) \\ u_{im} &= P(x_i = 1|d_m). \end{aligned}$$

(Strictly speaking, the network corresponding to equation 6.1 should have a single node $\mathbf{x}$ in place of $x_1, \ldots, x_n$ since equation 6.1 makes no independence assumptions. Independence is, however, assumed in all derivations based on equation 6.1 so we have chosen to show it in the network.)

Using the same notation and variables, the network of figure 4.1 can be reduced to the network of figure 6.3. This inference network is described by the probability distribution

$$\begin{aligned} P(R, f_k, x_1, \ldots, x_n, d_m) &= P(R|d_m) \\ &= P(R|f_k) P(f_k|x_1, \ldots, x_n) P(x_1|d_m) \ldots P(x_n|d_m) P(d_m). \end{aligned}$$

Comparing figure 6.3 with figure 6.2 we see that in the inference network model the query does not appear as a separate prior (root) but is explicitly conditioned on the representation concepts. Again, d_m is given, so we have

$$P(R|d_m) = P(R|f_k) P(f_k|x_1, \ldots, x_n) P(x_1|d_m) \ldots P(x_n|d_m).$$

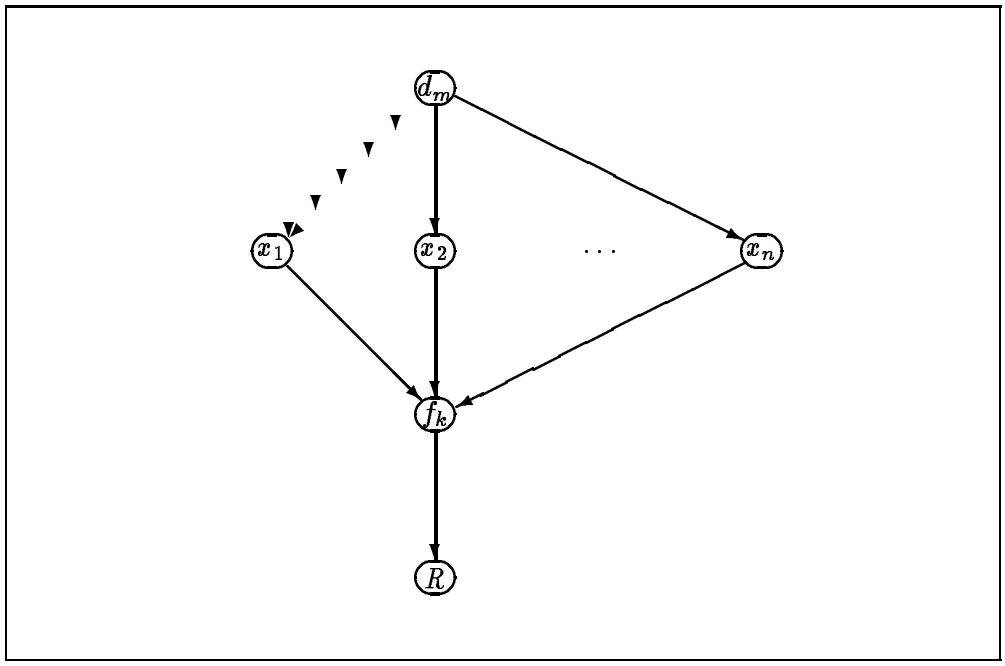

Figure 6.3: Example inference network

Applying Bayes' rule we get

$$P(R|d_m) = P(R|f_k)\frac{P(x_1,\ldots,x_n|f_k)P(f_k)}{P(x_1,\ldots,x_n)}P(x_1|d_m)\ldots P(x_n|d_m).$$

Assuming that the x_i are distributed independently in documents (6.3) and that the assignment of the x_i is independent of the query (6.4), that is,

$$P(x_1,\ldots,x_n) = \prod_{i\leq i\leq n} P(x_i) \tag{6.3}$$

$$P(x_1,\ldots,x_n|f_k) = \prod_{i\leq i\leq n} P(x_i|f_k) \tag{6.4}$$

we have

$$P(R|d_m) = P(R|f_k)P(f_k)\sum_{\mathbf{x}\in X}\prod_{1\leq i\leq n}\frac{P(x_i|f_k)}{P(x_i)}P(x_i|d_m). \tag{6.5}$$

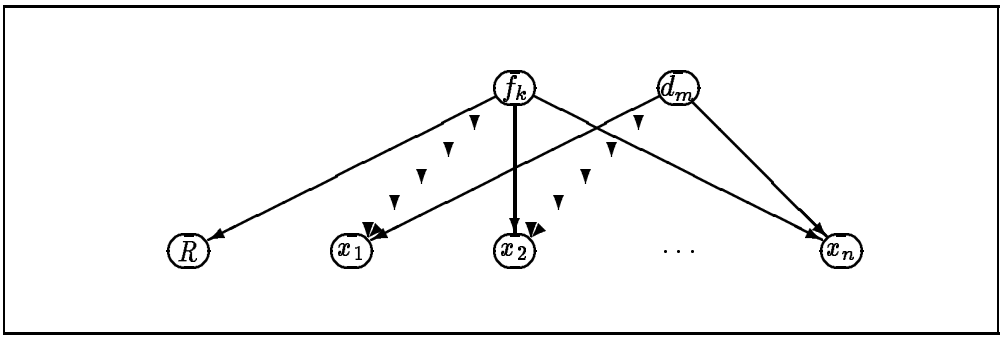

Figure 6.4: Effect of inversion

The application of Bayes' rule essentially inverts the network of figure 6.3 to obtain the equivalent network shown in figure 6.4[3]. Note that the use of Bayes' rule here is to allow us to derive a closed-form ranking expression that can be compared with the RPI model. In practice, we would use an estimate of $P(f_k|x_1,\ldots,x_n)$ and would not invert the network.

Equation 6.5 reduces to

$$P(R|d_m) = P(R|f_k)P(f_k)\prod_{1\le i\le n}\left(\frac{P(x_i=1|f_k)}{P(x_i=1)}P(x_i=1|d_m) + \frac{P(x_i=0|f_k)}{P(x_i=0)}P(x_i=0|d_m)\right).$$

If we let

$$\begin{aligned} p_{ik} &= P(x_i=1|f_k) \\ q_i &= P(x_i=1) \end{aligned}$$

[3]While the networks in figures 6.3 and 6.4 are equivalent in the sense that the computed probability distributions are the same, figure 6.4 does not lend itself to normal belief network updating procedures. In order to produce the new $P(x_i|f_k,d_m)$ link matrix and the new prior $P(f_k)$ we must make use of the assumed value of $P(d_m)$. In essence, when we invert the network we fold the prior probability of d_m into the new link matrix and extract a new prior for the query. This means that to test the effect of a change in $P(d_m)$, we would have to recompute the link matrices at each x_i and compute a new $P(f_k)$. With the network in figure 6.3, we can change our assumed value for $P(d_m)$ without changing the probability information stored at each node.

$$u_{im} \quad = \quad P(x_i = 1|d_m)$$

we get the ranking expression

$$P(R|d_m) = P(R|f_k)P(f_k) \prod_{1 \leq i \leq n} \left(\frac{p_{ik}}{q_i} u_{im} + \frac{1 - p_{ik}}{1 - q_i}(1 - u_{im}) \right) \tag{6.6}$$

Equation 6.6 differs from equation 6.2 in that p_{ik} is conditioned only on the query and not on R and the resulting probability is normalized by $P(f_k)$. The difference in conditioning for p_{ik} arises because the network of figure 6.3 implicitly assumes that $\mathbf{x}$ and R are conditionally independent given the query, that is, $\mathbf{x}$ cannot influence our assessment of relevance except through its effect on the query. The network of figure 6.2 assumes that $\mathbf{x}$ and f_k are independent, but not necessarily conditionally independent given R, that is, $\mathbf{x}$ and the query can influence our assessment of relevance independently. Under the assumption of conditional independence

$$P(\mathbf{x}|R, f_k) = P(\mathbf{x}|f_k)$$

and the p_{ik} terms are identical. $P(f_k)$ is constant for a given query and does not affect the ranking so, under the assumption of conditional independence, the rankings produced by the two models are identical.

The networks in figures 6.2 and 6.3 help to clarify the differences between the conventional probabilistic and the inference network retrieval models. In the network of figure 6.2, the query is modeled as a separate variable that is related to the possible document descriptions through the specification of $P(R|\mathbf{x}, f_k)$. The network of figure 6.3 explicitly models the dependence of the query on the document representation and the dependence of relevance on the query. Again, the network of figure 6.3 asserts the independence of the document representation and relevance given the query; the document representation cannot influence the probability of relevance except through its influence on the query.

The principal difference between the two models, then, lies in the dependencies assumed. While we have chosen Fuhr's model as the basis for comparison, network forms could be developed for the many other probabilistic formulations. The chief advantage of the inference network model is that it allows complex dependencies to be represented in an easily understood form and it allows networks containing these dependencies to be evaluated without development of a closed form expression that captures these dependencies.

6.2 Comparison with Unified Model

Two influential formulations of the probabilistic model are due to Maron, Kuhns, and Cooper [MK60, CM78] (generally referred to as the Maron and Kuhns model) and to Robertson and Sparck Jones [RS76, van79] (the binary independence model of Section 6.1.1). A third formulation which generalizes these two models was proposed by Robertson, Maron, and Cooper [RMC82]. This unified model clarifies two different interpretations of relevance within the probabilistic framework and provides an interesting contrast to the inference network model. For the remainder of this section, we use representation concept to refer to either a document representation concept or a query representation concept.

6.2.1 Maron and Kuhns model

The focus of the Maron and Kuhns model is probabilistic indexing, that is, given a set of documents $d_1, \ldots, d_n$, and a set of representation concepts $r_1, \ldots, r_s$, how do we estimate the probability that a user submitting a single representation concept as a query will judge a given document to be relevant. For clarity, this model restricts attention to single term queries.

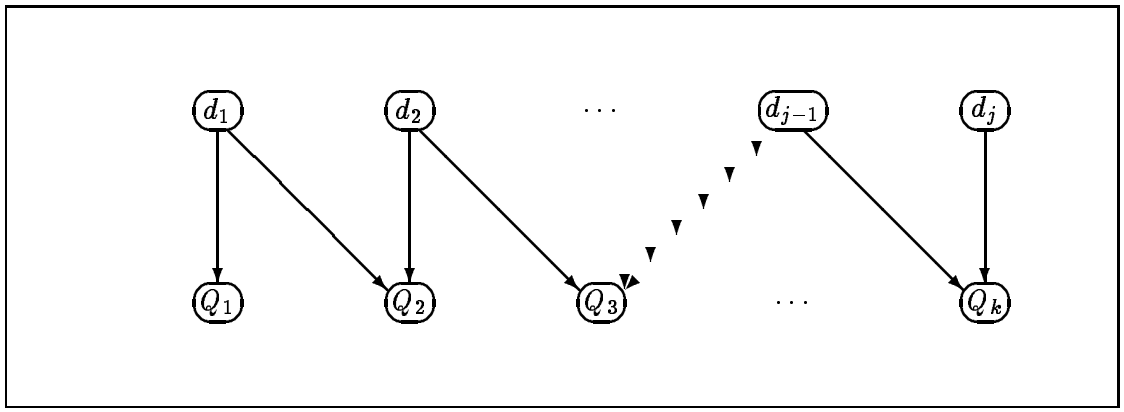

Figure 6.5: Network for Maron and Kuhns model

The approach advocated by Maron and Kuhns was to keep track of the number of times each query is submitted and which documents are judged relevant and non-relevant to each query. This information is then used to determine the frequency with which each document in the collection has been judged relevant to each query submitted. This frequency information is then used to estimate the probability of relevance and documents are ranked accordingly. This model, then combines the judgements of multiple users to compute the probability of relevance with respect to a set of equivalent queries.

This probability of relevance is independent of any information about which concepts occur in documents. In terms of the inference network model, these representation concepts are the query concepts contained in user queries and the document representation concepts play no part in estimating the probability of relevance. This gives the network shown in Figure 6.5 in which we learn the full link matrix form for each query concept based on a set of sample queries. Since queries are restricted to single terms, when processing a query we simply pick the appropriate representation node, instantiate each document, and rank documents by the belief at the representation node. No separate query network is required.

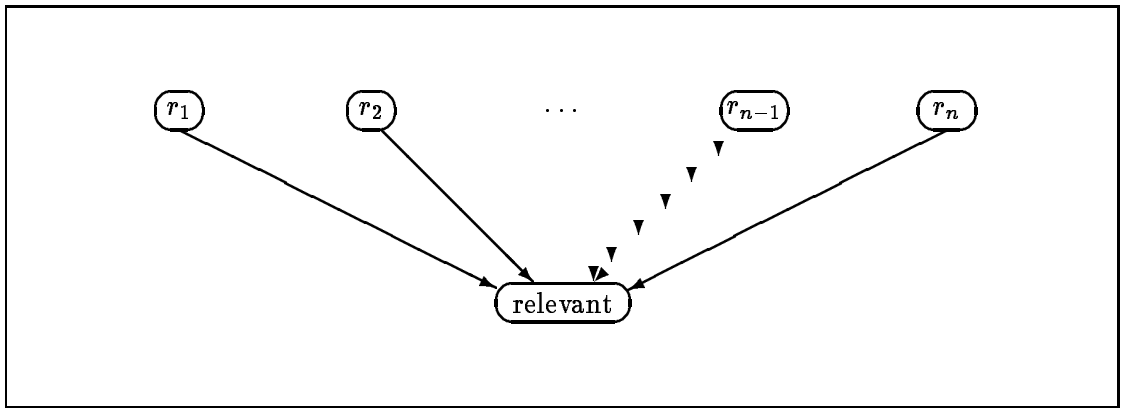

Figure 6.6: Network for Robertson and Sparck Jones model

6.2.2 Robertson and Sparck Jones model

The Robertson and Sparck Jones model takes a much different approach. We again maintain a history of queries and relevance judgements, but here we view the association between documents and representation concepts as fixed by the collection and independent of the use of these representation concepts in queries (although users can only submit as queries those concepts that have been associated with documents). In the Maron and Kuhns model the association between a representation concept and a document is created by a relevance judgement; in the Robertson and Sparck Jones model the association is known from characteristics of the individual documents and documents appear in the model only as the set of representation concepts assigned to them. In the Robertson and Sparck Jones model, a query is a set of one or more of these representation concepts and the objective of the retrieval system is to compute the probability that a randomly selected document is relevant given a set of representation concepts as a query. Given appropriate independence assumptions, we can compute this probability for each concept separately and combine them using a canonical rule.

In inference network terms, the representation concepts of the Robertson and Sparck Jones model are the document representation concepts. This approach gives the network of Figure 6.6 (which repeats Figure 6.1) in which the documents and query concepts are omitted and a single *relevance* node is attached directly to the document representation concepts. The link matrix at the *relevance* node is learned from the training sample under a set of independence assumptions.

6.2.3 Unified model

To integrate these two views, the unified model considers documents and groups of similar query *uses*, where a query use is the submission of a query by a user with an information need and a group of similar uses is the set of all uses for a single query. If we let

$$\begin{aligned} A &= \text{the class of all (past and future) query uses,} \\ C &= \text{the class of all (present and future) documents,} \\ b_k &= \text{a single query use,} \\ d_m &= \text{a document,} \end{aligned}$$

then the event space for the unified model is $A \times C$ and relevance is defined as a relation $R \subseteq A \times C$ where a document/use pair $(b_k, d_m) \in R$ if and only if d_m would be judged relevant to b_k. We now define a partitioning $B = B_1, \ldots, B_n$ of the set A into sets of similar query uses (all uses for a single query), and a partitioning D of the set C into sets containing similar documents (a component set of D will have more than one element only if two documents are assigned exactly the same set of representation concepts). Using this notation, the probability of relevance under the Maron and Kuhns model is $P(R|B_i, d_m)$, that is, relevance is defined

for a single document and the set of all uses of a given query. Relevance for the Robertson and Sparck Jones model is $P(R|B_k, D)$, that is, relevance is defined for a single query use and all documents. For the unified model, relevance is defined for a single document and a single query use, that is, $P(R|b_k, d_m)$.

6.2.4 Comparison of the unified and inference network models

The unified model is similar to the inference network model in that it distinguishes between document and query representations (document and need properties in Robertson, Maron, and Kuhns terms) and views the problem of estimating the probabilities for the two representations as separate.

The two models differ, however, in at least four important respects. First, in the network model there is no notion of a query use. The notion of a query use arises as a result of the Maron and Kuhns interpretation of representation concepts as queries and the use of a sample of relevance judgements from several uses to estimate the probability of relevance. As will be discussed later, we question whether this kind of learning is practical without a number of additional assumptions about the structure of the network. As a result, we believe that this kind of learning can best be modeled as one component of a procedure to estimate the link matrices at the query nodes and that the retrieval model need not explicitly represent query uses.

Second, the inference network model represents documents, document representation concepts (document properties), queries, and query representation concepts (query properties) as separate variables. The development of the unified model at various times requires that we assume that queries are single terms or that we introduce auxiliary variables to represent document properties. This difference between the two models is notational, but we think it clearer to explicitly represent all of the variables in the model.

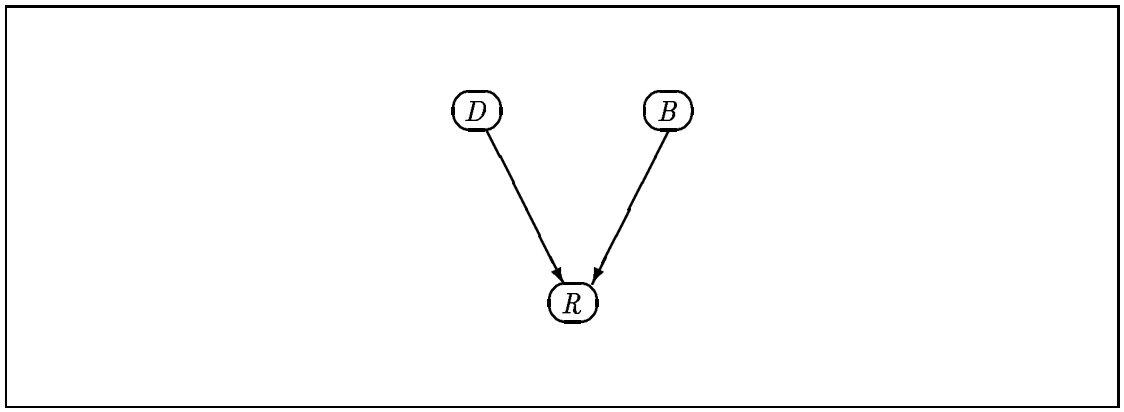

Figure 6.7: Network for the unified model

Third, the network model uses an explicit dependence structure in which belief in document representation concepts causes belief in query representation concepts which, in turn, causes belief in the information need. This assumption implies, among other things, that while belief in document and query concepts both cause belief in the information need, they do not act independently. The unified model is less clear here, but the main independence assumption that can be derived from the model is that query and document representation concepts are independent, that is

$$P(b_k, d_m) = P(b_k) \cdot P(d_m). \tag{6.7}$$

Robertson, Maron, and Kuhns explore the consequences of additional independence assumptions, but no firm conclusions about an appropriate set are reached. Assuming that a dependence among these variables (documents, query uses, and relevance) does exist, equation 6.7 suggests the network of Figure 6.7 for the unified model which, like the RPI model, assumes that the document and query representations are independent causes of our belief in relevance. While it is tempting to posit a set of document nodes above the sets of document representations (D) and query representation (B) in Figure 6.7, this would assert a dependence between

the representations *unless* we have evidence establishing the value of document nodes with certainty (i.e., they are all fully instantiated).

The final difference between the two models lies in a philosophical difference regarding the nature of probability estimates; the unified model is frequentist while the inference network model takes a modified subjectivist view (see [Sav54] for a discussion of the history and nature of frequentist versus subjectivist interpretations).

The structure of the unified model is influenced by the desire to estimate the required probabilities based on observed frequencies. This orientation gives rise to the notion of query uses and definition of the event space in terms of all past and future queries and all present and future documents.

We take the view that estimates based on frequency data are to be preferred when they are available but that, for these kinds of models, we will generally not have representative samples and that other techniques will be required to estimate the probabilities. In operational settings, the vast majority of queries will never have been seen before, even after a long period of data collection. Further, those queries that have occurred more than a few times may well not have the same underlying information need. Under these conditions, it is difficult to view the samples that will be available in practice as representative. Robertson, Maron, and Cooper are clearly concerned with this problem and suggest some subjectivist techniques for estimating the probabilities (e.g., using thesaurus information). They also suggest that techniques for estimating single term weights from multiple term query samples might be possible (e.g., [FB90]).

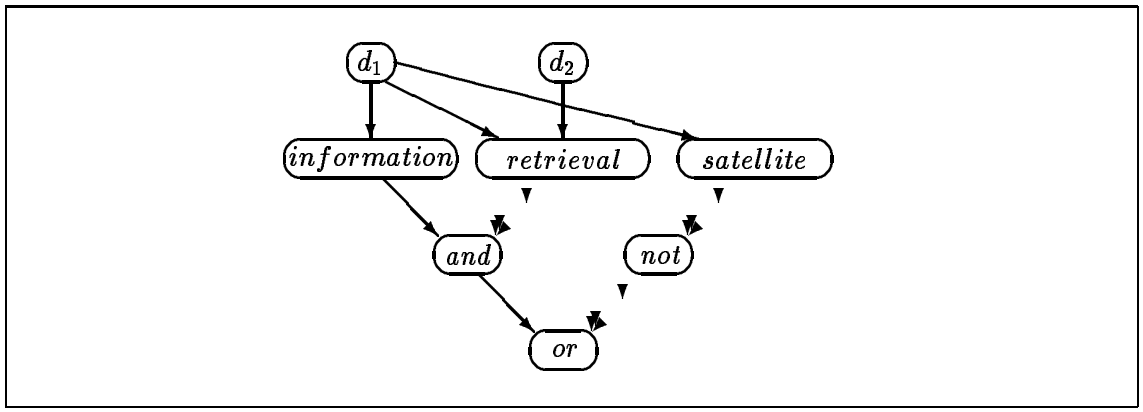

Figure 6.8: Inference network for $(information \wedge retrieval) \vee \neg satellite$

6.3 Boolean retrieval

Inference networks can be used to precisely simulate Boolean retrieval and they provide a natural interpretation of the semantics of Boolean operations in probabilistic terms. In this section we first show how Boolean retrieval can be simulated and then show how the probabilistic interpretation of the Boolean operations can be relaxed to produce document rankings.

6.3.1 Implementing Boolean retrieval

Using the canonical link matrix forms of section 4.5 we can simulate Boolean retrieval as follows. For clarity, we assume that the query and representation vocabularies are identical so we can omit query concepts from the network.

1. Use a canonical *or* matrix at each representation node. When a document is instantiated, all representation concepts to which it has been attached will have $\text{bel}(r_i) = 1$. All remaining representation concepts have $\text{bel}(r_j) = 0$.

2. Build an expression tree for the query. The root of the tree is the query and all arcs in the tree are directed toward the root. The leaves of this tree will be representation concepts and the interior nodes will correspond to expression operators. At each operator node use the canonical link matrix form for that operator. Attach this tree to the document network. A simple example for the query

   `(information and retrieval) or not satellite`

   is shown in figure 6.8 (the use of **`or not`** is somewhat unusual, most commercial systems support only **`and not`**).

3. Using the evaluation procedure described in section 4.3, instantiate each document in turn and record the belief in the query node. Any document for which $\text{bel}(Q) = 1$ satisfies the query, any node for which $\text{bel}(Q) < 1$ does not.

Under the assumptions above and using binary indexing, $\text{bel}(Q)$ can only have values 0 or 1 and the inference network simulates a conventional Boolean system exactly.

The same probabilistic interpretation of the Boolean operators applies equally well to weighted indexing. Using the approach described in section 4.5.1 we can incorporate indexing weights by replacing the *or* link matrix in the representation concept nodes with a *wtd_sum* matrix incorporating the appropriate *tf* and *idf* weights. In this case, when a document is instantiated, all representation nodes to which it is attached take on the *tf.idf* weight for that term/document pair and all remaining representation nodes take on a node-specific default belief, for now we assume that all default beliefs are 0. These weights are then combined using the closed-form expressions of section 4.5. In short, the *tf.idf* weights are

interpreted as probabilities and are combined using the normal rules for negation and for disjunctive or conjunctive combination of sets in an event space. As a result, the inference network model provides a natural probabilistic interpretation of the Boolean operators and of indexing weights.

An interesting result of the inference network research is an efficient implementation of a general *not* operator. Commercial systems allow only an *and not* operator since *or not* and unary *not* generally produce unwieldy retrieved sets. The inference network model efficiently supports general evaluation of *not*.

Conventional Boolean retrieval systems use inverted files in which each term has an associated list of documents in which it occurs. When evaluating a simple Boolean expression, two lists are combined to create a new result list. The lists associated with most terms are quite short (again, by Zipf's law, roughly half are of length one) and query evaluation is generally optimized to keep the size of intermediate results as short as possible. As an example, let a be a common term that occurs in 5% of the documents in a collection of n documents and let b be a term that occurs in a single document. Evaluation of the expressions a *and* b or a *or* b requires that the inverted lists for a and b be read and results in a new list that is no longer than the list for a. An expression like a *or not* b, however, will probably require reading a list containing nearly every document in the collection and will create a result list containing at least $n - 1$ documents. For a collection containing several million documents, this is a very expensive operation. The problem with *or not* and unary *not*, then, is that they turn very short inverted lists into very long inverted lists.

As will be discussed in Chapter 9, retrieval inference networks are also implemented using inverted files. Each representation concept has an associated list of parent documents with the belief in the representation concept that results when

each parent document is instantiated. Evaluation of *not b* requires only that we use equation 4.9 to recompute belief for the single document in *b*'s inverted list. Computing *a or not b*, then, has nearly the same cost as computing *a or b*.

6.3.2 Relaxing the interpretation of Boolean operators

The binary nature of the retrieval decision in Boolean systems is frequently cited as a drawback [Cro86, SM83, Sal88, LB88]. Intuitively, we would like a document containing all but one term of an *n*-term *and* to be judged nearly as likely to match the query as a document containing all n terms and substantially more likely to match than a document containing none of the terms. The binary decision arises because of our strict probabilistic interpretation of the Boolean operators. The link matrix forms we have chosen assert complete certainty given the evidence available at a node, that is

$$\begin{aligned} P(Q_{and} = \text{true} | \text{all parents} = \text{true}) &= 1 \\ P(Q_{or} = \text{true} | \text{any parent} = \text{true}) &= 1 \\ P(Q_{not} = \text{true} | \text{parent} = \text{true}) &= 0. \end{aligned}$$

We can easily relax this interpretation of the probabilistic semantics of the Boolean operators. Simply reducing the certainty associated with the operators, that is,

$$\begin{aligned} P(Q_{and} = \text{true} | \text{all parents} = \text{true}) &= c_a \\ P(Q_{or} = \text{true} | \text{any parent} = \text{true}) &= c_o \\ P(Q_{not} = \text{true} | \text{parent} = \text{true}) &= c_n \end{aligned}$$

$0 \leq c_a, c_o, c_n \leq 1$ does not have the desired effect since it does not incorporate our intuition about the significance of the number of parents having a particular

value. It simply compresses the range of values for bel(Q). A better approach is to choose a value $n \leq c \leq \infty$ where n is the number of parents at a given node and to interpret the *and* operator to mean

$$\begin{aligned} P(Q_{and} = \text{true} | n \text{ parents} = \text{true}) &= 1 \\ P(Q_{and} = \text{true} | k \text{ parents} = \text{true}) &= 1 - \frac{n-k}{c}, \quad 0 < k < n \\ P(Q_{and} = \text{true} | \text{no parents} = \text{true}) &= 0 \end{aligned}$$

and the *or* operator to mean

$$\begin{aligned} P(Q_{or} = \text{true} | n \text{ parents} = \text{true}) &= 1 \\ P(Q_{or} = \text{true} | k \text{ parents} = \text{true}) &= \frac{k}{c}, \quad 0 < k < n \\ P(Q_{or} = \text{true} | \text{no parents} = \text{true}) &= 0 \end{aligned}$$

Under this interpretation, when $c = \infty$ the operators have their normal Boolean interpretation. As c decreases, our belief in Q depends increasingly on the number of parents that are true. When $c = n$ the distinction between *and* and *or* has disappeared, the link matrices for both operators are the same, and both are equivalent to the *sum* link matrix of section 4.5. Note that since a node implementing the *not* operator has exactly one parent, its interpretation is unchanged. These interpretations for the Boolean operators can be implemented as canonical link matrices requiring $O(1)$ space and operating in $O(n)$ time.

The use of this parent weighting scheme is quite similar to the extended Boolean retrieval or p-norm model [Sal88, SFW83]. In this model, the similarity between a document (represented as a binary vector $D = (d_1, d_2, \ldots, d_n)$) and a query Q consisting of the conjunction or disjunction of all terms in the vector is given by

$$\text{Sim}(D, Q_{and}) = 1 - \left[\frac{(1-d_1)^p + (1-d_2)^p \ldots + (1-d_n)^p}{n} \right]^{\frac{1}{p}}$$

$$\text{Sim}(D, Q_{or}) = \left[\frac{d_1^p + d_2^p \ldots + d_n^p}{n}\right]^{\frac{1}{p}}$$

When $c = n$ and $p = 1$, both models produce the same results. For a conjunctive query containing n terms, m of which are true, under the p-norm model we have

$$\begin{aligned}\text{Sim}(D, Q_{and})_{p=1} &= 1 - \left[\frac{(1-d_1)^p + (1-d_2)^p \ldots + (1-d_n)^p}{n}\right]^{\frac{1}{p}} \\ &= 1 - \frac{n-m}{n}, \quad 1 \leq m \leq n\end{aligned}$$

and for the network model we have

$$P(Q_{and} = t | m \text{ parents true}) = 1 - \frac{n-m}{n}, \quad 1 \leq m \leq n.$$

For a disjunctive query using the p-norm model we have

$$\begin{aligned}\text{Sim}(D, Q_{or})_{p=1} &= \left[\frac{d_1^p + d_2^p \ldots + d_n^p}{n}\right]^{\frac{1}{p}} \\ &= \frac{m}{n},\end{aligned}$$

and using the network model we have

$$P(Q_{or} = t | m \text{ parents true}) = \frac{m}{n}.$$

Similarly, when $c = p = \infty$ both models produce the same results

$$\text{Sim}(D, Q_{and})_{p=\infty} = P(Q_{and} = t | m \text{ parents true}) = \begin{cases} 1 & \text{if } m = n \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Sim}(D, Q_{or})_{p=\infty} = P(Q_{or} = t | m \text{ parents true}) = \begin{cases} 0 & \text{if } m = 0 \\ 1 & \text{otherwise} \end{cases}$$

For values $n < c < \infty$ and $1 < p < \infty$, both functions are monotonic in the number of true parents (number of non-zero vector elements) m, but they are not equivalent since $P(Q = t)$ is linear in m while $\text{Sim}(D, Q)$ is not. Note that we could choose to

redefine our probability function to produce equivalent results. If, for example, we used

$$c = \sqrt[p]{\frac{n-m}{n}}$$

for *and* operations, then the models would produce results that are equivalent in the sense that, for any value of p we can find a corresponding value of c that produces the same values for bel(Q) and Sim(D, Q). There is, however, no theoretical basis for this redefinition.

The inference network model handles weighted indexing as a natural extension and is again equivalent to the p-norm model for $p = 1$ and $p = \infty$ and is similar but not equivalent for $1 < p < \infty$.

The performance of the network and p-norm models will be compared in Chapter 8.

6.4 Estimating the probabilities

Given the link matrix forms of section 4.5, we now consider the estimates required for the basic model of figure 4.1. The only roots in the inference network of figure 4.1 are the document nodes and the prior probability associated with these nodes is set to 1/(collection size). Estimates are required for five different node types: text, representation and query concepts, query, and information need.

Text nodes. Since text nodes are completely dependent upon the parent document node, the estimate is straightforward. Since there is a single parent, a matrix form can be used; t_i is true exactly when d_i is true and false exactly when d_i is false so

$$L_{\text{text}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

This matrix form is the inverse of that used for *not*.

Note that the distinction between document and text nodes is not required for the basic model and we often ignore text nodes for clarity. Text nodes are required if we support sharing of text by documents. If we allow document nodes to share text nodes, then an *or* matrix at the text node is appropriate, t_i is true when any parent is instantiated. Link matrices for advanced features are discussed in Chapter 7.

Representation concept nodes. Link matrix forms for representation concepts were discussed in section 4.5. For binary indexing and unweighted terms an *or*-combination can be used so that $P(r_i = true) = 1$ if any parent is instantiated. Link matrix forms that incorporate *tf*, *idf*, and *tf.idf* weights are described in Section 8.2.2.

Query concept nodes. As we have seen, previous research on indexing and term weights can be incorporated directly in the document network. The query network, particularly the links between representation and query concepts, is less well understood. Here we are interested in estimating the dependence of concepts mentioned in the user's query upon the representation concepts. Most current retrieval models view these two sets of concepts as identical under the assumption that the user knows the set of representation concepts and can use them to formulate queries. Research suggests, however, that the mismatch between query and indexing vocabularies may be a major cause of poor recall [FLGD87, Tur90]. While our current implementation assumes a one-to-one correspondence between query and representation concepts, it would appear that improved estimates of the dependence of query concepts on representation concepts could markedly improve performance. Two areas of research bear directly on improving the quality of these estimates: automatic thesaurus construction and natural language research aimed at extracting concept descriptions from query text, identifying synonymous or related descriptions, and resolving ambiguity [LC90, KC89].

Query nodes. The dependence of query nodes on the query concepts is more straightforward. For Boolean queries we can build an evaluation tree using the link matrix forms described in section 6.3; we can adjust link matrix values if we have information about the relative importance of the query concepts. For probabilistic queries we can use a weighted-sum matrix. Strategies for setting the query concept weights are described in Section 8.2.1.

Information need. The information need can generally be expressed as a small number of queries of different types (Boolean, m-of-n, probabilistic, natural language, ...). These can be combined using a weighted-sum link matrix with weights adjusted to reflect any user judgments about the importance or completeness of the individual queries.

Chapter 7

Extensions to the basic model

The basic model described in chapter 4 is limited in at least two respects. First, it does not describe how relevance feedback can be incorporated. Second, we have represented only a limited number of dependencies between variables. In this chapter we will see that these limitations can be removed.

7.1 Feedback

There are two basic ways in which feedback can be incorporated in an inference network: adding evidence or altering the dependencies represented in the network. The two approaches are fundamentally different. Adding evidence always leaves the probability distribution represented in the network unchanged but alters beliefs in the network to be consistent with that distribution. Altering the dependencies, either by changing the topology of the network or by altering the link matrices changes the underlying probability distribution which in turn alters belief. The use of evidence is appropriate when we know that the distribution is "correct" (if, for example, the topology is known and the link matrices have been learned from a reliable sample). Evidential feedback is appropriate in the document network

which is largely determined by the characteristics of the collection. Frisse and Cousins [FC89] use this approach to implement feedback in a hierarchy of index terms associated with a hypertext medical handbook.

Altering dependencies is appropriate when the initial network is known to be an approximation to the correct distribution and we obtain better information about the nature of the true distribution. This approach is used in document space modification [YM88, FB90] where we use a set of queries and relevance judgements to learn the "correct" distribution for documents and representation concepts. This approach can also be used in the query network which changes as we gain information about the user's information need. This new information can be derived from many sources, but we will restrict attention to the traditional sources of relevance feedback information, user judgements about the relevance of sample documents.

We will adopt the second approach, then, and will view relevance feedback is a process through which we alter and refine the structure of the query network. Essentially, given a sample of relevant documents we wish to develop one or more new query representations that will retrieve documents similar to those in the relevant sample. These new query representations can either augment or replace the original representation. Traditional probabilistic relevance feedback can be readily adapted to the network model. It may be possible to further improve performance, however, if we use a Boolean or hybrid query form in place of the *sum* used in traditional relevance feedback.

7.1.1 Probabilistic feedback

In conventional probabilistic retrieval, documents are ranked using a (generally) linear discriminant function in which each term corresponds to a representation

concept in the collection. Typically, only the representation concepts found in the query have non-zero values and the coefficients of these terms are estimated using some model-specific function. A representative function is

$$g(d) = \sum_i (\log \frac{p_i}{1-p_i} + \log \frac{1-q_i}{q_i}) \tag{7.1}$$

where p_i is the probability that term i occurs in a relevant document and q_i is the probability that term i occurs in a non-relevant document. The second term in the summation is typically estimated using each term's *idf* and the first term is initially estimated using some fixed p_i (e.g., $p_i = 0.5$) or based on the frequency of the term in the query.

In the network model, query terms are represented by establishing conditional dependence relationships between the query node and the appropriate representation concept nodes and the coefficients appear as weights in the link matrix at the query node ($P(Q|r_i)$).

In relevance feedback we are given a sample of documents that have been judged relevant and we wish to re-estimate our linear discriminant function based on this sample. In a sense, the set of relevant documents is used to augment (or replace) the original query (it is also possible to use negative feedback based on a set of retrieved documents that were judged nonrelevant, but we will consider only positive feedback). In a typical relevance feedback strategy, we would compute a new set of p_i values for equation 7.1 based on the relevant sample and either add the top n terms to the original query terms or simply replace the original query with the top n terms to produce a new linear discriminant function.

In the network model we can use exactly the same strategy. We add links between the query node and the representation concepts to be added and re-estimate the link matrix weights based on the sample of relevant documents rather than on

the query text. Whatever estimation strategy is used to produce an estimate for p_i can be incorporated directly in the link matrix weights.

Similar relevance feedback strategies (e.g., negative feedback or fixed-increment correction [van79]) can be simulated using the same basic feedback model.

7.1.2 Boolean feedback

A number of models have been proposed for using relevance feedback with Boolean retrieval systems [Sal88, SVF84, Rad88, Rad83]. While some of these models have been shown to significantly improve performance when compared to conventional Boolean retrieval, they are not attractive in the network environment. These models generally adapt probabilistic relevance feedback techniques to estimate weights for terms in very restricted Boolean query forms (e.g., disjunctive normal form with no negation and *and* terms containing at most three representation concepts). Since these models do not make use of any linguistic or domain knowledge, it is unlikely that they will afford performance gains that cannot be achieved with normal probabilistic relevance feedback.

The development of an effective relevance feedback mechanism for Boolean queries is a potentially important area for further research. Encoding feedback information in a Boolean query could improve performance more than as a probabilistic query since it is possible to encode information in the Boolean query that is not representable in the *sum* expression. A Boolean feedback mechanism could be implemented with the following procedure:

1. Identify important words and phrases in the sample of relevant documents. Important words can be identified using the same techniques as for probabilistic feedback, but phrases are a harder problem. Work on the identification of

syntactic or statistical phrases [LC90] should be of use here and it may be useful to ask the user about the importance of candidate words and phrases.

2. Identify words and phrases whose absence from the set of relevant documents is important. This is a hard problem since the sample of relevant documents is usually far too small to allow meaningful statistical estimates. An expert system may be of use here if the domain of interest is known, but for the near term, the user is probably the only reliable source of information about concepts that should not be in the relevant set.

3. Estimate weights for the concepts identified above, probably using both statistical analysis and user input.

4. Assemble a query. An obvious approach would be to use *and* (or a new proximity operator) to define phrases, use *or* to assemble sets of concepts that should and should not be in relevant documents, and then use *and not* to combine the two sets of concepts.

A number of variation are possible. In particular, initial work suggests that the probabilistic *sum* operator might be more appropriate than the *or* in step 4. This would represent a hybrid approach in which we start with the basic probabilistic relevance feedback query and selectively add Boolean structure in the form of phrases and negation.

The main reason for considering the use of Boolean expressions for feedback is to increase the expressive power of the feedback query without incurring the expense of learning a complete link matrix. Clearly, once the set of representation concepts is known we could learn the appropriate link matrix given a large enough sample of relevant documents. Unfortunately, we will rarely have a large enough sample

and the computational costs associated with learning a "correct" matrix would be prohibitive if more than a few terms were involved.

7.2 Additional dependencies

In the basic model, we assume that there are no dependencies between documents, between texts, between representation concepts, between query concepts, or between queries. While independence assumptions like these are not uncommon in retrieval models, it is widely recognized that the assumptions are unrealistic; there are a number of both statistical and logical dependencies between representation concepts and between documents. In particular, we would like to incorporate term and document clustering and would like to represent citation links between documents and thesaurus relationships between terms.

As before, we identify the set of nodes upon which a given node depends and characterize the probability associated with each node conditioned on its immediate parents. When adding these new links, however, we must be careful to preserve the acyclic nature of the inference network. Bayesian inference networks cannot represent cyclic dependencies; evidence attached to any node in a cycle would continually propagate through the cycle and reinforce the original node. In the basic model, no cycles are possible since nodes are only linked to node types that are lower in the DAG. The introduction of these "horizontal" dependencies makes cycles possible.

7.2.1 Document and term clustering

A variety of clustering techniques have been developed to improve information retrieval performance [van79]. These may be loosely categorized as *document* clus-

tering techniques which attempt to divide the collection into (possibly overlapping) subsets which are similar and *term* clustering techniques which attempt to identify subsets of representation concepts with similar usage or meaning. Clustering techniques differ widely in the document or term attributes considered, the definition of a similarity or dissimilarity measure, and the structure of the resulting classification. Term clustering techniques represent one kind of automatically-built thesaurus in which terms contained in a cluster are, in some sense, synonymous; these thesaurus clusters may be organized in a hierarchy to represent broader and narrower classifications. Representation of these thesaurus-like relationships will be discussed in Section 7.2.3.

Document clustering techniques are used to retrieve documents that are similar to a relevant document under the assumption that similar documents are related to the same queries. Our use of cluster information is somewhat unusual since we do not retrieve clusters; we incorporate the cluster information in the dependence relationships between document texts and representation concepts. In the network fragment shown in Figure 7.1, document texts t_1, t_2, and t_3 are indexed using representation concepts r_1, r_2, r_3, and r_4. Documents t_2 and t_3 have been identified as part of cluster c_1; both texts are linked to a cluster node and the cluster node is linked to the representation concepts that define the cluster. The cluster node is similar to a conventional cluster representative. Documents t_1 and t_2 are indexed by the same representation concepts (r_1 and r_2) and, if we assume equivalent conditional probabilities, would be ranked equivalently in the absence of the cluster node. With the addition of the cluster node, however, a new representation concept (r_3) is associated with t_2 by virtue of its cluster membership. Assuming that r_3 contributes positively to the belief in q, t_2 would be ranked higher than t_1. Like query nodes, cluster nodes are a representation convenience, it is always possible

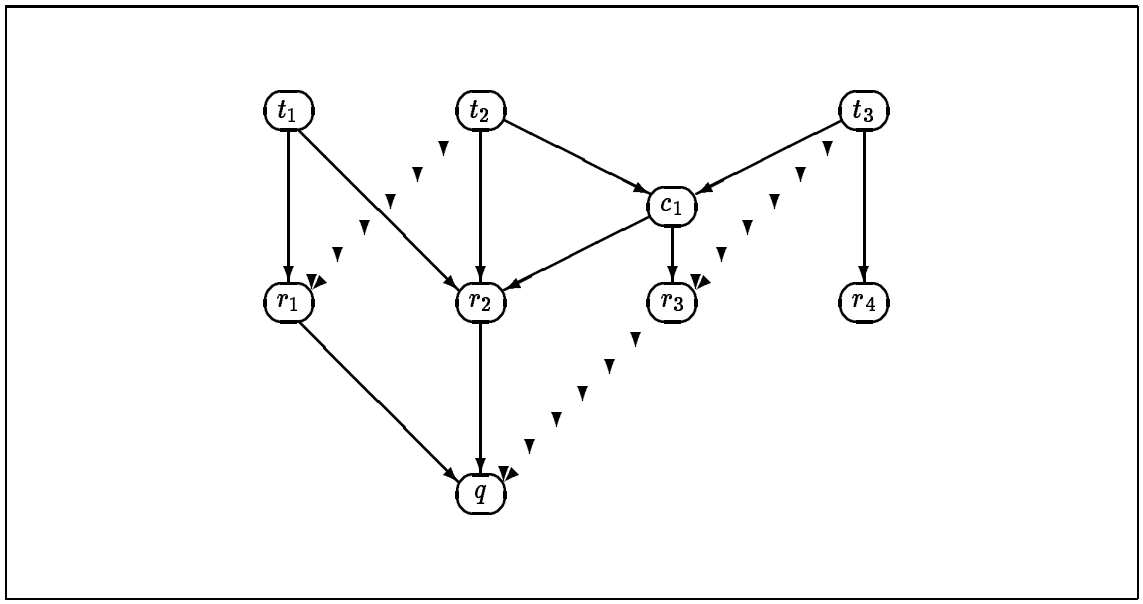

Figure 7.1: Document clustering model

to eliminate them by increasing the complexity of the distribution specified at the representation concept nodes.

7.2.2 Citation and nearest neighbor links

A variety of asymmetric relationships between pairs of documents can also be represented. These relationships are similar to clustering in that they use similarity between documents to expand the set of representation concepts that can be plausibly associated with a text. They differ in that they are ordered relations defined on pairs of documents rather than an unordered, set membership relationship between documents and clusters. The existence of one of these links can be treated as new evidence about the set of concepts that should be assigned to a document. The use of evidence here rather than a direct link between documents avoids the possibility of cyclic dependencies that could arise with these directed links.

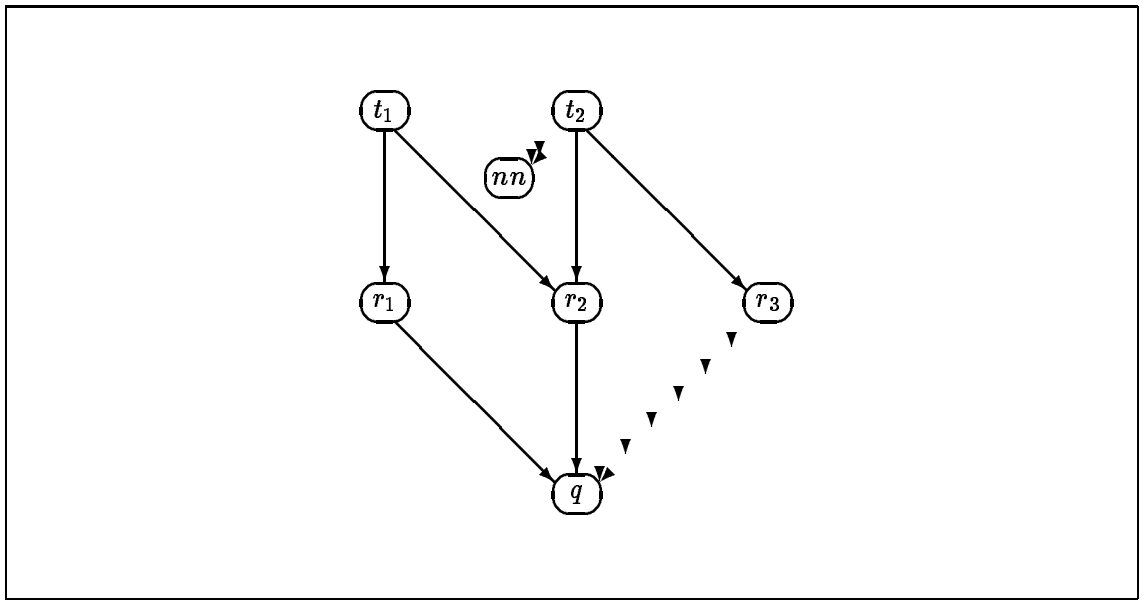

Figure 7.2: Nearest neighbor link

Perhaps the best example of this kind of relationship is the nearest neighbor link in which a document is linked to the document judged to be most similar to the original document. In Figure 7.2 the set of representation concepts associated with document t_1 is expanded by virtue of its nearest neighbor link to document t_2 which is represented as an evidence node attached to t_2 when t_1 is instantiated. A second kind of ordered link is based on citations occurring in the text. Citation links may be useful if the type of reference can be determined (e.g., citing a similar work, a peripherally related work, or a work presenting an opposing viewpoint) to allow estimation of the probabilistic dependence between the nodes. Nearest neighbor and citation links are discussed in detail in Sections 8.6.1 and 8.6.2.

7.2.3 Thesaurus relationships

The structure of these networks provides a natural mechanism to represent probabilistic dependencies between the concepts or terms that describe documents and information needs. These relationships are similar to conventional thesaurus relationships, but include more information. For example, a conventional thesaurus might list "house pet" as a broader term for "dog" and "cat"; the network representation will include a specification of the probability that "house pet" should be assigned given a document containing "dog" or "cat" in isolation, neither term, or both terms.

In this discussion, we restrict attention to common thesaurus relations (synonym, related term, broader term, and narrower term). We also deal only with the use of this information for simple automated inference about document and query content. We do not deal with frame-structured thesauri that attempt to capture the detailed structure of domain knowledge [SSG89] or with the use of a thesaurus to aid to the user during query formulation [Pol87].

Synonyms can be represented in the network by creating a node to represent an equivalence class and adding it as a child to each synonym in the equivalence class. This approach can also be used for "near" synonyms or related terms where the belief that the equivalence class has been observed is dependent upon the presence of a specific combination of representation concepts. Building these equivalence classes into the document network is probably not useful for two reasons. First, their presence could represent a significant computational burden. This is especially true since the number of potential equivalence classes is very large, but only a few will be useful for a given query [ST84]. Second, it would be necessary to consult an external thesaurus when building the query network to determine when to attach a

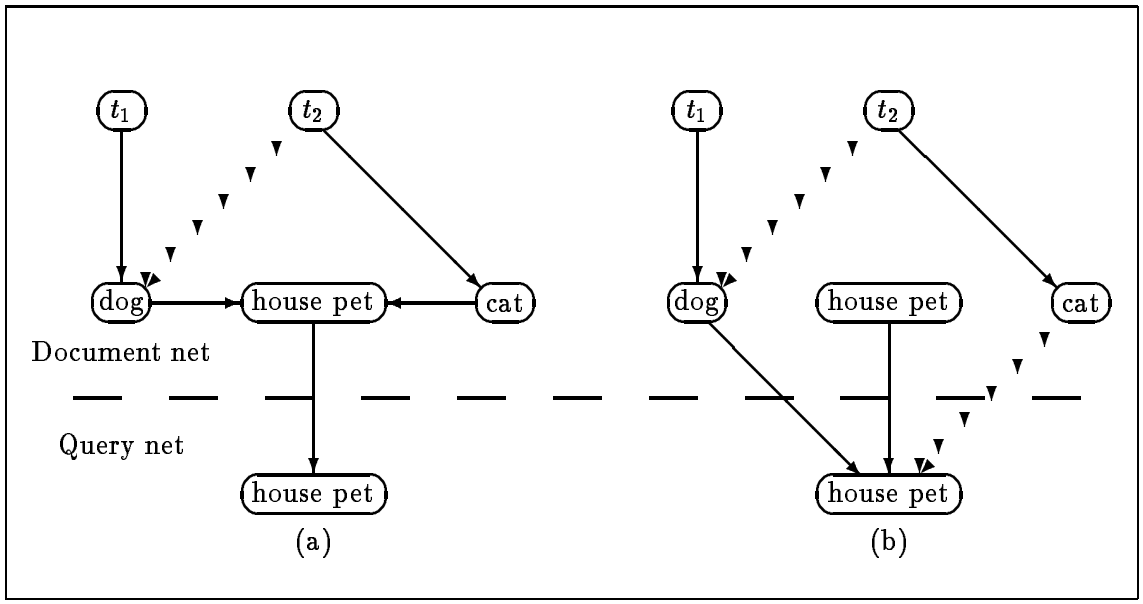

Figure 7.3: Broader term representation

query concept to the equivalence class node rather than a representation concept. Since a query concept node represents exactly this kind of equivalence class, the external thesaurus should be used to handle synonyms and closely related terms as the query network is built.

Broader term relationships are very similar to the relationship between synonyms and their equivalence class. They differ mainly because a broader term is generally a member of the set of representation concepts. As with synonyms we could represent the relationship in the document network (Figure 7.3a), but will generally prefer to represent it in the query network (Figure 7.3b).

Synonyms, related term, and broader term relationships are useful primarily as a means of determining the relationship between the query concepts used to describe the information need and the representation concepts that describe document content. Narrower term relationships are somewhat less useful for inferring vocabulary

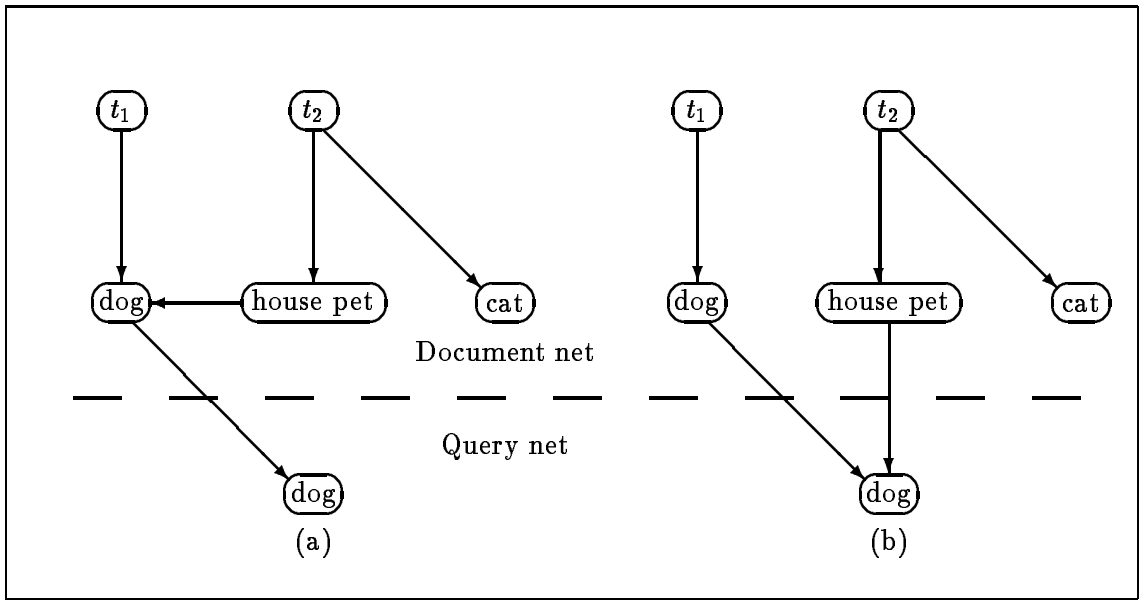

Figure 7.4: Narrower term representation

relationships. For example, if representation concepts "dog" and "cat" have been assigned to a document, we can infer that a set of broader terms (e.g., "house pet" or "mammal") could plausibly be assigned to the same document. However, the fact that "mammal" has been assigned to a document represents fairly weak evidence that "cat" should also be assigned.

Direct representations of narrower terms in the network is possible (Figure 7.4a), but, again, we will generally prefer to represent them in the query network (Figure 7.4b).

7.2.4 Phrases

The use of phrases as representation concepts provides a much more detailed representation of document content than do individual terms. The use of phrases in information retrieval is an area of active research, see [Lew90] for a review.

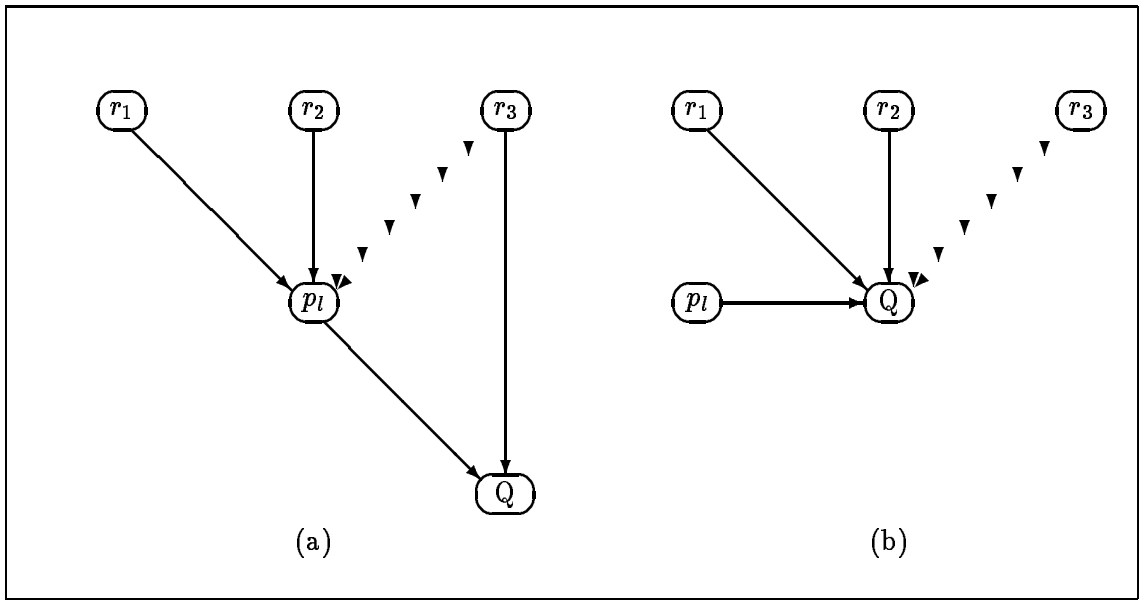

Figure 7.5: Representing phrase dependencies

One advantage of the network model is that it allows explicit representation of the dependence between phrases and their component terms.

Figure 7.5 shows the two basic ways in which phrases can be added. In Figure 7.5a the phrase p_l is added as a new representation concept that is independent of the component terms (r_i, r_j, and r_k). A document containing the phrase will also contain all three terms, but this dependence is not represented in the document network. A query which contains the phrase is linked to p_l and optionally to the component terms. Figure 7.5a is similar to the phrase model developed by Croft [Cro86] in which the presence of a set of dependent terms (a phrase) results in an additive correction to the independence model ranking function. It would be possible to represent the dependence between the terms and the phrase in the link matrix at Q (i.e., r_i, r_j, and r_k have less influence on belief in Q when p_l is observed),

but this would require a detailed knowledge of the relationships between terms and phrases in the collection.

Figure 7.5b shows a model which represents the dependence of the phrase on its component terms. In this model the phrase represents a concept that can be inferred based on the presence of the parent terms. In a simple form of this model, we might infer the presence of the phrase only when all three parent terms are present and the combining function at p_l is an *and*. In general, however, we may believe that p_l is an accurate description of a document even when all of the parent terms are not present. Our degree of belief may depend on the specific set of parent terms that are observed and the combining function is a full link matrix. Other combining functions are certainly possible, in this context a weighted *and* canonical form might be useful. Experimental results using a simple *and* model are discussed in Chapter 8.

Recent work suggests that it may be possible to learn the appropriate combining function. [FCAT90] describes procedures for learning link matrices given a set of documents and basic qualitative dependence information.

With either phrase model we must still determine which concepts are contained in the query in order to make the appropriate connections between the query and document networks. In general, it will be possible to connect the query to both the phrase and to the component terms. In Figure 7.5b, attaching to the phrase and a term means that the two concepts both occur in the query. In Figure 7.5a, multiple attachments may also arise if it is necessary to represent the dependence between a phrase and its component terms.

In the basic inference network model we use information about the presence or absence of a term in the document and the characteristics of a term's use in the collection to estimate belief in representation concepts. For phrases it will probably

be useful to consider additional sources of evidence (e.g., syntactic relationships between terms and term proximity) when estimating belief in a phrase.

7.3 Rule based inference

Document retrieval inference networks can also be used to support the kind of inference normally associated with deductive databases if we interpret the operators of deductive logic in terms of conditional probability. Material implication as the interpretation of if-then rules in deductive databases is replaced in Bayesian nets by an interpretation based on conditional probability, that is, a rule $a \rightarrow b$ is interpreted as a probability $P(b|a)$. The normal interpretations of *and*, *or*, and *not* can be simulated using canonical link matrix forms if we assume that terms are independent (as is customary with deductive logic). Given a node q with parents $p_1, \ldots, p_n$, for *and*-combinations $((p_1 \wedge \ldots \wedge p_n) \rightarrow q)$ we are interested in computing $p(q|p_1, \ldots, p_n)$. Under our assumption of independence, this is simply the product of the individual conditional probabilities

$$p(x|p_1, \ldots, p_n) = p(x|p_1) \cdot p(x|p_2) \cdot \ldots \cdot p(x|p_n)$$

which can be computed using the current belief and link matrix values. For *or*-combinations we are interested in computing the probability of q given the set of $p_1, \ldots, p_n$. Again under the assumption of independence, this computation can be done using a composite link matrix which combines the effect of the individual parents (see [Pea88] for a detailed treatment of "noisy OR-gates").

Figure 7.6 shows an example with three document representations and a small set of rules. Two content predicates are shown: `about(document,term)` asserts that a given document representation can be described by the term. The

```
about(d1,'deductive database')      about(d2,'algorithms')
about(d1,'logic and databases')     about(d2,'languages')
about(d1,'null values')             about(d2,'capture rule')
about(d1,'relational database')     about(d2,'query language')
about(d1,'query language')          about(d2,'relational database')
                                    about(d2,'Prolog')
cites(d3,d1)
cites(d3,d2)
about(d3,'logic queries')
about(d3,'relational database')
about(d3,'horn clause')
about(d3,'logic database')

about(di,ti)∧ cites(dj,di)→ about(dj,ti)
about(di,ti)∧ synonym(ti,tj)→ about(di,tj)

about(di,'Prolog')→ about(di,'logic programming')
about(di,'query language')→ about(di,'database')
synonym('logic database','deductive database')
```

Figure 7.6: Deductive document database fragment

$$\texttt{cites(document}_1\texttt{,document}_2\texttt{)}$$

predicate asserts that a reference to $\texttt{document}_2$ was found in $\texttt{document}_1$. Additional predicates represent knowledge about the relationships between terms or between document representations (e.g. `synonym('logic database','deductive database')` or a nearest neighbor predicate $\texttt{nn(d}_i\texttt{,d}_j\texttt{)}$ which asserts that the representation of $\texttt{d}_i$ is "close" to that of $\texttt{d}_j$.).

The inference rules fall into two basic classes. General rules describe inferences that are independent of a specific database and express general knowledge about the relationship between predicates. For example, $\texttt{about(d}_i\texttt{,t}_i\texttt{)} \wedge \texttt{synonym(t}_i\texttt{,t}_j\texttt{)} \rightarrow \texttt{about(d}_i\texttt{,t}_j\texttt{)}$ allows us to infer that any document characterized by a term is also characterized by a synonym for that term. Domain specific rules represent knowledge about the meaning of the terms in a given database, for example

$$\texttt{about(d}_i\texttt{,'prolog')} \wedge \texttt{about(d}_i\texttt{,'programming')} \rightarrow$$
$$\texttt{about(d}_i\texttt{,'logic programming')}.$$

A query represents a theorem to be proven and has the form $q = \{d_i | W(d_i)\}$ where W is a first-order logic expression which represents the constraints that a document must satisfy in order to be retrieved. This query form is more restrictive than typically used in a deductive database since only document representations may appear in the answer to a query rather than an arbitrary set of free variables (one can imagine queries like $\{\texttt{t}_i | \texttt{about(d}_i, \texttt{t}_i) \wedge \texttt{date(d}_i, \texttt{1900})\}$, to collect all terms assigned to documents published in 1900, but this kind of query is not generally supported in conventional retrieval systems).

Conventional queries translate directly into logic expressions. For example, a query "find documents about deductive databases and Prolog" is represented as the theorem

$$\texttt{about(d}_i\texttt{,'deductive databases')} \wedge \texttt{about(d}_i\texttt{,'Prolog')}$$

The deductive database approach allows expression of other kinds of queries that are not generally supported by conventional document retrieval systems, for example $\texttt{cites(d}_i\texttt{,d}_j\texttt{)} \wedge \texttt{author(d}_j\texttt{,'Smith')}$ might be used to retrieve all documents that cite any document written by Smith.

While the deductive approach to document retrieval offers some attractive advantages, it has two major shortcomings: it does not provide any mechanism for dealing with uncertainty and does not provide a suitable mechanism for combining multiple sources of evidence. Both of these shortcomings are addressed in the current retrieval model.

Chapter 8

Results

In this chapter, we first review the objectives and hypotheses established for this research (Section 8.1). We then present results for the basic model which includes estimating the required probabilities (Section 8.2) and results for simple searches (Section 8.3), multiple query representations (Section 8.4), and multiple document representations (Section 8.5). Finally, we present results for the extended model in which citations and nearest neighbor clusters are used as additional sources of evidence (Section 8.6) and summarize the major research results (Section 8.7).

8.1 Objectives and hypotheses

The original objectives established for this research were to:

1. Determine the retrieval performance achievable under the inference network model.

2. Compare the retrieval performance of the inference network model with that obtained with a conventional probabilistic model using the same searches run on the same databases.

3. Compare retrieval performance within the inference network model of different document representations.

4. Compare retrieval performance within the inference network model of different query representations (natural language, Boolean, and both).

5. Compare the computational performance of these networks with that of conventional probabilistic models and to characterize the costs associated with these networks as a function of network size.

These objectives led to the following hypotheses.

1. Given equivalent document representations, query forms, and assumptions about the match between indexing and query vocabularies, the inference network model will perform as well as probabilistic models.

2. The use of networks containing multiple document representations will significantly improve retrieval performance when compared to equivalent networks without the additional representations.

3. The use of multiple query formulations and search strategies will significantly improve retrieval performance when compared to equivalent networks with a single natural language query.

   Hypotheses 2 and 3 deal with the features of the basic model in which no dependencies between documents or between terms are represented.

4. The incorporation of dependencies between documents (citations and nearest neighbor links) will significantly improve retrieval performance when compared to equivalent networks that do not incorporate these additional dependencies.

5. It will be possible to build and evaluate these networks in "reasonable" time. For a collection that contains t term occurrences we will interpret "reasonable" to mean a) $O(t^2)$ time to build and preevaluate the network, b) query evaluation time relatively independent of document network size and less than 10 seconds for 90% of the queries when run on a typical micro-Vax or Sun network, and c) storage overhead that is a small constant ($c < 5$) times the original collection size.

The results for Hypotheses 1 through 4 are reported here. Hypothesis 5 is discussed in Chapter 9.

8.2 Estimating the probabilities

For the basic model we must provide two probability estimates: one estimate characterizes the dependence of a query or information need upon the terms in the collection (Section 8.2.1) and the second characterizes the dependence of individual terms on their parent documents (Section 8.2.2).

In what follows, we attempt to discuss the individual components of these estimates independently, but they are, in fact, dependent. As a result, conclusions about the performance of one component cannot be based on a single experiment. We will generally choose a baseline set of parameters and show the effect of varying an additional parameter. Except where noted, results represent trends that hold over a broad range of parameters, but actual performance changes will vary with the baseline selected.

As a simple example, experiments (discussed in section 8.2.2.2) show that using a non-zero estimate for the probability that a term should be assigned to a document in which it does not occur (a *default* probability) consistently improves retrieval performance. The amount of improvement depends on a number of factors in addition to the estimate used for the default. As shown in Table 8.1, adding a "reasonable" default estimate to a network in which no query weights are used gives a very significant improvement (20.6%) in average precision. Adding the same default estimate to a network in which *tf.idf* query weights are used produces only a modest improvement (3.3% with most of the improvement occurring at high recall levels). If we base our conclusion on a network with unweighted query

Table 8.1: Effect of default estimate on retrieval performance

	Precision (% change) – 50 queries			
	Unweighted query terms		Weighted query terms	
Recall	No default	With default	No default	With default
10	52.8	62.7 (+18.9)	67.3	67.2 (−0.1)
20	45.9	51.9 (+13.0)	55.2	56.1 (+1.7)
30	37.1	41.4 (+11.7)	47.2	47.9 (+1.4)
40	26.9	33.6 (+24.9)	38.8	39.3 (+1.2)
50	22.6	27.6 (+22.0)	33.0	32.9 (−0.4)
60	17.1	23.2 (+35.6)	26.1	27.8 (+6.6)
70	11.9	16.5 (+39.1)	19.1	21.4 (+11.8)
80	9.5	12.8 (+33.8)	15.4	17.3 (+12.2)
90	5.2	6.3 (+20.1)	10.4	11.9 (+14.1)
100	3.3	4.2 (+29.4)	8.2	9.6 (+16.5)
average	23.2	28.0 (+20.6)	32.1	33.1 (+3.3)

terms we would conclude that adding a default estimate improves performance with a significance level of 0.001. If we base our conclusion on a network with weighted query terms the significance level is only 0.031. We have attempted to use representative baselines throughout and will point out results that are unusually dependent on the choice of a baseline.

In order to reduce the number of tables we try to group related results in a single table. Unfortunately, this means that some results for combined queries (i.e., natural language and Boolean) are presented before multiple query representations are discussed (in Section 8.4). Given that the use of multiple query representations has been discussed in Chapter 4, this ordering is preferred to presenting the same baseline results in two separate tables.

8.2.1 Dependence of queries on the document network

As suggested in section 4.5, the procedures for estimating $P(Q|t_i)$ differ for Boolean and probabilistic queries. Estimates for Boolean queries use product-form estimates

which are largely determined by the model, while estimates for probabilistic queries use weighted sums and are based on the results of previous information retrieval research.

8.2.1.1 Probabilistic queries

A probabilistic query is based on a natural language description of an information need. Words in the query that generally do not affect retrieval performance (stop words) are removed and the remaining words are stemmed to remove common endings in an attempt to reduce simple spelling variations to a single form. Documents are then ranked using some function that combines the scores for each document term that occurs in the set of query terms. A number of these combining functions have been used (see [MKN79] for a review), but the most common technique is to simply add the individual term scores after some weighting function has been applied. This weighting function is intended to increase the influence of terms that are believed to be important on the final ranking. The weighted-sum link matrix of section 4.5 implements precisely the desired function; we need only specify the weights to be associated with query terms.

Two factors are commonly used in weighting the contribution of query terms – the frequency of the term in the query (qf) and the inverse document frequency (idf) of the term in the collection. The basic ideas are that 1) a content-bearing term that occurs frequently in the query is more likely to be important than one that occurs infrequently, and 2) those terms that occur infrequently in the collection are more likely to be important than frequent or common terms. The measure used for within-query frequency is simply the raw frequency of the term in the query.

The *idf* used is a standard *idf* definition that has been normalized to produce a value in the range [0..1]. The *idf* measure is given by

$$idf = \frac{\ln(\frac{\text{collection size}}{\text{term frequency}})}{\ln(\text{collection size})}. \tag{8.1}$$

Singly-occurring terms have an *idf* score of 1.0 and *idf* scores decrease as the frequency of a term in the collection increases. From Zipf's law we expect that roughly half of the terms in the collection will occur only once and will have an *idf* score of 1.0. Based on the information in Table 2.1, *idf* scores lie in the range $0.109 \leq idf \leq 1.0$ for both the CACM and CISI collections.

Table 8.2: Effect of query weighting on retrieval performance (CACM)

	Precision (% change) – 50 queries			
Recall	no weights	*idf* weights	*qf* weights	*qf.idf* weights
10	63.2	62.6 (−1.0)	67.7 (+7.2)	67.2 (+6.4)
20	52.2	51.3 (−1.8)	54.5 (+4.4)	56.2 (+7.7)
30	45.6	43.0 (−5.9)	48.3 (+5.9)	47.6 (+4.2)
40	35.9	34.7 (−3.3)	41.4 (+15.2)	41.3 (+14.8)
50	30.3	28.6 (−5.7)	35.6 (+17.3)	34.4 (+13.6)
60	24.2	24.7 (+2.1)	27.1 (+12.0)	29.1 (+20.5)
70	16.9	16.8 (−0.4)	19.4 (+15.3)	20.3 (+20.6)
80	13.7	13.3 (−2.6)	15.8 (+15.6)	16.3 (+19.4)
90	7.8	8.2 (+3.9)	10.8 (+37.4)	11.3 (+44.4)
100	5.7	5.7 (+0.8)	8.3 (+47.3)	8.7 (+54.3)
average	29.6	28.9 (−2.3)	32.9 (+11.3)	33.3 (+12.5)

Table 8.2 shows the retrieval performance obtained with no query term weights, query terms weighted by *idf* only, query terms weighted by within-query frequency only, and query terms weighted by both factors. The weighting function used is given by equation 4.10, essentially, each term score (belief) is multiplied by the appropriate term weight and the sum is normalized by the maximum achievable score given the term weights.

The use of *idf* weights alone decreases retrieval performance and the performance loss is somewhat more pronounced in the high precision half of the ranking. Within-query weights (qf) and the combination of qf and *idf* weights both increase retrieval performance (significance levels of 0.002 for both weightings). The combined weighting shows somewhat higher average precision, but most of this gain is at high recall levels and performance actually drops somewhat at high precision levels. For these experiments, the *qf.idf* and qf weightings are not significantly different. In general, the performance levels of qf and *qf.idf* weightings are quite similar, and the choice of one over the other will depend on the function used for $P(t_i|d_j)$ and on the query types to be used. For many applications, the qf weighting will be preferred since it is simpler and performs at least as well as *qf.idf* at high precision. Unless otherwise noted, all results for probabilistic queries use either qf or *qf.idf* weighting.

The relative performance of these four query term weightings is consistent across a wide range of ranking functions, but the actual performance depends heavily on the $P(t_i|d_j)$ estimate used. The comparison Table 8.2 is relatively conservative since it uses one of the best estimates developed for term belief as the baseline. As suggested in Table 8.1, much more dramatic improvements could be demonstrated if a less effective baseline was used.

8.2.1.2 Boolean queries

A Boolean query consists of an expression using the operators *and*, *or*, and *not* with query terms as operands. Stopwords are removed from the queries, query terms are stemmed as with probabilistic queries, and documents are ranked using equations 4.1 through 4.9 to combine probabilities.

For conventional Boolean queries, then, the retrieval model specifies how probabilities are to be combined and no additional probabilities need be estimated. Since it would be straightforward to add weights to terms in Boolean queries, a open research issue is whether significant improvements can be obtained by weighting Boolean query terms. While the Boolean queries available with the standard test collections do not include term importance information, [CD90] suggests that useful information about term importance can be readily obtained from users; this information could be directly incorporated as weights for Boolean expression evaluation.

It is also possible that some form of *idf* weighting could be used to improve performance of Boolean queries or that weighting strategies developed for conventional Boolean systems [NKM77] could be adapted to the network model. An experiment in which each intermediate *and* and *or* expression was weighted using an *idf* based on the number of documents in the intermediate result did not improve performance.

8.2.2 Dependence of term belief on documents

The probability that a term accurately describes the content of a document can be estimated in several ways, but previous information retrieval research has consistently shown within-document frequency (tf) and inverse document frequency (idf) to be useful components of such estimates [SB87]. In developing estimates we concentrated on functions involving tf and idf; other functions are certainly possible and could be used in the basic model. The idf measure used in term belief estimates is that given in equation 8.1.

8.2.2.1 Estimating the *tf* component

Two different measures were used for within-document frequency in the experiments. The first is the common normalized tf [SM83, van79] in which the tf score

for term i in document j is given by

$$ntf_{ij} = \frac{\text{frequency of term } i \text{ in doc } j}{\text{max frequency for any term in doc } j}.$$

This measure gives a very broad range of values ($0.037 \leq ntf \leq 1.0$ for both CACM and CISI). Intuitively, this range is too broad. A term that occurs once in a document in which the maximum *tf* is 20 is probably not 1/20 as important as the most frequently occurring term. The fact that a term occurs at all is a significant event and the frequency of occurrence (> 1) is of secondary importance. In order to test the effect of a non-linear *tf* estimate, a second function, given by

$$nltf = \frac{1.0 + \ln(f_{ij})}{1.0 + \ln(\max f_{ij})}$$

was used. This function compresses the range of belief values slightly ($0.233 \leq nltf \leq 1.0$) and increases belief more for small term frequencies than for term frequencies near the maximum. Log normalization was not used for query term weighting since the range of frequencies queries is quite small (a term rarely occurs more than twice in a query).

Table 8.3 shows the performance of the two estimates in the same ranking function for the CACM collection. For probabilistic queries, the log normalized estimate is better at a significance level of 0.063. There is, however, very little difference for Boolean queries (no difference for one set of queries, raw frequency normalization better at a significance level of 0.500 for the other) or when query forms are combined (raw-frequency normalization better at a significance level of 0.125).

When used with the CISI collection, the raw-frequency normalization produced slightly better results for probabilistic and combined queries (significance level of 0.500 for both) and equivalent results for Boolean queries (Table 8.4). Given these

Table 8.3: Raw-frequency versus log-frequency normalization with CACM

	Precision (% change) – 50 queries					
	Probabilistic		Boolean		Combined	
Recall	*ntf*	*nltf*	*ntf*	*nltf*	*ntf*	*nltf*
10	65.8	67.6 (+2.7)	64.8	64.8 (−0.1)	74.3	76.5 (+3.0)
20	54.7	55.6 (+1.6)	56.8	58.2 (+2.4)	64.4	65.5 (+1.6)
30	45.9	46.0 (+0.2)	49.2	48.7 (−1.0)	56.6	55.4 (−2.1)
40	39.5	39.1 (−1.0)	44.0	44.3 (+0.8)	49.8	48.6 (−2.5)
50	33.7	33.4 (−0.9)	38.3	37.6 (−1.7)	44.8	42.3 (−5.5)
60	27.9	28.3 (+1.5)	33.3	32.2 (−3.3)	37.8	36.1 (−4.4)
70	19.8	21.1 (+7.0)	22.4	22.7 (+1.3)	25.6	25.5 (−0.4)
80	16.0	17.5 (+9.3)	17.8	17.6 (−1.3)	21.4	21.1 (−1.4)
90	11.0	12.3 (+11.8)	11.3	10.9 (−4.1)	13.5	12.7 (−5.3)
100	8.6	9.9 (+14.7)	7.8	7.6 (−2.4)	10.4	9.6 (−7.3)
average	32.3	33.1 (+2.4)	34.6	34.5 (−0.3)	39.9	39.3 (−1.3)

results, the simpler raw-frequency normalization should probably be preferred for most applications, although either estimate works well and performance does appear to be collection dependent, at least for probabilistic queries. Experiments with additional collections will be required to determine if one estimate is superior.

Given these basic forms for the *tf* and *idf* estimates, several experiments were conducted to determine

1. what range of belief values for a term is appropriate given that a term occurs in an instantiated document ($P(t_i|d_j = \text{true})$),

2. what range of belief values is appropriate given that a term does not occur in an instantiated document ($P(t_i|d_j = \text{false})$), and

3. how the *tf* and *idf* components should be combined when forming the overall estimate of term belief.

Strictly speaking, our use of $P(t_i|d_j = \text{true})$ and $P(t_i|d_j = \text{false})$ here is a bit loose. In the basic model, only one parent document is instantiated at a time so that any

Table 8.4: Raw-frequency versus log-frequency normalization with CISI

	Precision (% change) – 50 queries								
	Probabilistic			Boolean			Combined		
Recall	*ntf*	*nltf*		*ntf*	*nltf*		*ntf*	*nltf*	
10	38.2	35.6	(−6.8)	45.0	45.4	(+0.8)	44.7	42.4	(−5.2)
20	28.1	27.0	(−4.1)	33.1	32.9	(−0.5)	34.6	34.3	(−0.9)
30	20.3	19.4	(−4.6)	24.9	24.5	(−1.5)	25.3	25.2	(−0.1)
40	17.0	17.0	(−0.3)	20.7	20.2	(−2.1)	20.3	20.5	(+0.9)
50	15.3	15.0	(−2.3)	18.0	17.8	(−1.2)	17.9	17.5	(−2.3)
60	12.7	12.6	(−0.9)	15.0	14.8	(−1.2)	15.1	14.7	(−2.2)
70	11.1	11.0	(−1.0)	12.6	12.7	(+0.9)	12.5	12.8	(+2.5)
80	8.8	8.8	(−0.4)	9.4	9.4	(+0.3)	9.5	9.5	(−0.7)
90	6.7	6.8	(+2.2)	6.9	7.0	(+1.5)	6.9	7.0	(+1.0)
100	4.5	4.5	(+0.8)	4.1	4.3	(+4.1)	4.5	4.6	(+2.9)
average	16.3	15.8	(−3.2)	19.0	18.9	(−0.3)	19.1	18.8	(−1.4)

term has either exactly one instantiated parent or no instantiated parents. When we write $P(t_i|d_j = \text{true})$ we really mean

$$P(t_i|d_j = \text{true} \wedge \text{all other parents=false}).$$

When we write $P(t_i|d_j = \text{false})$ we really mean

$$P(t_i|\text{all parents=false}).$$

When no confusion will arise, we will use the shorthand notation. We will also occasionally refer to $P(t_i|d_j = \text{true})$ as the belief in t_i given that parent d_j is true and will refer to $P(t_i|\text{all parents=false})$ as the default belief for t_i.

8.2.2.2 Establishing a range for $\mathbf{P(t_i|d_j = true)}$

In the network model, the absence of any information for or against a proposition is represented by a belief of 0.5. Our initial hypothesis, then, was that estimates

Table 8.5: Effect of varying belief threshold for probabilistic queries

	Precision – 50 queries			
Recall	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$
10	66.4	67.2	65.6	61.6
20	55.8	56.2	53.5	49.6
30	45.5	47.6	45.8	41.9
40	38.0	41.3	38.7	35.7
50	32.2	34.4	33.3	30.6
60	27.4	29.1	26.9	25.7
70	20.2	20.3	19.1	18.5
80	17.1	16.3	15.6	14.9
90	11.3	11.3	10.6	10.2
100	8.5	8.7	8.4	7.9
average	32.2	33.3	31.8	29.7

for $P(t_i|d_j = \text{true})$ should lie in the range 0.5 to 1.0 and estimates for the default belief should lie in the range 0.0 to 0.5.

The retrieval performance for a typical belief function [SM83, van79] given by

$$P(t_i|d_j = \text{true}) = \alpha + (1 - \alpha) * ntf * idf$$

where $P(t_i|d_j = \text{false})$ is fixed is shown in Table 8.5 for probabilistic queries and in Table 8.6 for Boolean queries. Retrieval performance peaks for $\alpha = 0.4$ which gives estimates for $P(t_i|d_j = \text{true})$ in the range 0.42 to 1.0, close to the predicted range. For larger values of α the range of scores is compressed and performance drops due to an increase in the number of tied documents. For smaller values of α, performance drops off sharply as estimates for $P(t_i|d_j = \text{true})$ begin to interfere with estimates for $P(t_i|d_j = \text{false})$. The low end performance drop can be reduced somewhat by lowering the estimate for $P(t_i|d_j = \text{false})$, but this in turn lowers overall performance.

These results are largely independent of the specific belief function used. While overall retrieval performance varies between functions, most appear to peak when

Table 8.6: Effect of varying belief threshold for Boolean queries

	Precision – 50 queries			
Recall	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$
10	68.3	67.6	67.0	65.5
20	58.8	58.8	57.4	56.0
30	49.7	50.2	49.1	48.5
40	45.4	45.2	43.5	43.0
50	39.9	39.8	38.5	37.4
60	33.3	33.6	33.2	32.6
70	21.8	22.0	21.2	20.8
80	16.5	18.1	17.5	17.3
90	10.5	11.4	11.2	11.0
100	7.7	7.9	7.6	7.6
average	35.2	35.4	34.6	34.0

the range of values for $P(t_i|d_j = \text{true})$ and $P(t_i|d_j = \text{false})$ do not overlap and the minimum value for $P(t_i|d_j = \text{true})$ is in the 0.4 to 0.5 range.

8.2.2.3 Default belief

Most probabilistic models form a document score by adding the scores for each term in common with the query; missing terms are treated as if they had a score of 0. In the inference network model, this amounts to asserting that the absence of a term in a document (in most cases, in the abstract or title of the document) implies certainty that the term should not be assigned to the document. Given the small size of the title and abstract compared to the full document and the fact that the query and document vocabularies are not identical, a less extreme estimate for the probability that a term should be assigned given that it is not observed seems more reasonable. This estimate should probably depend on the type of document record in use – the absence of a term from the full text of a journal article should be treated as stronger evidence than its absence from the abstract or title.

Table 8.7: Performance with fixed default estimates (NL queries on CACM)

	Precision (% change) – 50 queries					
Recall	0.0	0.1	0.2	0.3	0.4	0.5
10	64.9	66.0(+1.6)	67.1(+3.4)	67.8 (+4.5)	69.6 (+7.3)	62.0 (−4.5)
20	51.1	51.7(+1.4)	54.1(+6.0)	56.2(+10.1)	58.7(+14.9)	50.3 (−1.6)
30	43.3	44.4(+2.5)	46.1(+6.4)	47.7(+10.1)	48.8(+12.6)	38.2(−11.7)
40	35.9	37.1(+3.3)	38.6(+7.6)	41.9(+16.8)	42.4(+18.2)	31.5(−12.1)
50	30.8	32.0(+4.2)	33.3(+8.4)	35.8(+16.3)	34.2(+11.4)	27.2(−11.7)
60	25.7	26.9(+4.7)	27.4(+6.6)	29.1(+13.2)	28.8(+12.0)	23.5 (−8.4)
70	18.5	19.2(+3.5)	19.2(+3.7)	21.0(+13.2)	22.0(+19.0)	19.0 (+2.5)
80	14.9	15.3(+3.1)	15.8(+6.0)	17.1(+14.9)	18.5(+24.1)	15.4 (+3.6)
90	10.1	10.3(+2.7)	10.6(+5.0)	11.7(+16.6)	12.5(+23.9)	9.5 (−5.6)
100	7.9	8.1(+2.4)	8.4(+5.9)	9.3(+17.0)	9.6(+21.4)	8.0 (+1.2)
avg	30.3	31.1(+2.7)	32.1(+5.8)	33.8(+11.4)	34.5(+13.9)	28.5 (−6.1)

Since the within-document frequency for all of these term-document pairs is zero, we considered only estimates based on a term's collection frequency (*idf*) and hypothesized that the default belief should be proportional to the frequency of the term in the collection (inversely proportional to *idf*). Essentially, this hypothesis asserts that a common term is more likely to be a correct descriptor than a rare term given that neither occurs in the document.

Table 8.7 shows retrieval performance for probabilistic queries when a fixed default value is assigned to all terms (no *idf* weights). A higher default probability reduces the effect of a missing term on a document's ranking. It also compresses the range of belief values for the query given documents in the collection. Performance improves for default estimates up to about 0.3 or 0.4 and then falls off (again, this point can be increased by modifying the estimate for $P(t_i|d_j = \text{true})$, but this will not improve overall performance). Performance for Boolean queries is similar.

Since values in the 0.3 to 0.4 range produced the best retrieval performance, several default estimates of the form $\alpha - (\beta * idf)$ were tried, where α ranged from

Table 8.8: Use of *idf* in the default estimate (CACM collection)

	Precision (% change) – 50 queries					
	Probabilistic		Boolean		Combined	
Recall	fixed	*idf*	fixed	*idf*	fixed	*idf*
10	69.6	67.7 (−2.8)	70.0	69.0 (−1.4)	77.6	76.2 (−1.8)
20	58.7	56.5 (−3.7)	59.1	59.3 (+0.3)	65.1	66.3 (+1.8)
30	48.8	47.9 (−1.9)	50.1	50.7 (+1.1)	55.0	56.3 (+2.3)
40	42.4	42.8 (+0.9)	44.2	45.6 (+3.3)	47.5	50.9 (+7.3)
50	34.2	35.6 (+4.0)	38.2	39.8 (+4.3)	41.2	45.4 (+10.1)
60	28.8	30.0 (+4.1)	32.0	33.6 (+5.1)	36.1	38.8 (+7.3)
70	22.0	21.5 (−2.3)	21.9	22.2 (+1.1)	25.1	25.1 (−0.1)
80	18.5	17.7 (−4.3)	17.5	17.5 (+0.3)	19.3	20.4 (+5.3)
90	12.5	12.2 (−1.9)	11.0	11.4 (+4.0)	11.8	12.4 (+5.1)
100	9.6	9.5 (−1.5)	7.7	8.0 (+4.3)	8.2	9.2 (+11.1)
average	34.5	34.1 (−1.1)	35.2	35.7 (+1.6)	38.7	40.1 (+3.6)

0.2 to 0.5 and β ranged from 0.2 to 0.4 (estimates for $P(t_i|d_j = \text{true})$ were adjusted as necessary to avoid overlapping ranges). Table 8.8 shows the performance of a representative function with the CACM collection.

For probabilistic queries the *idf* default weighting does not improve performance when compared to the fixed default. For probabilistic queries in the CISI collection, however, the *idf* weighting improved performance somewhat (significance level of 0.250). For Boolean queries on both collections the *idf* weighting improves performance somewhat. For combined queries the *idf* weighting improves performance significantly for CACM (significance level of 0.016) and has little effect with the CISI collection. Given these mixed results, no conclusion can be drawn about the superiority of fixed versus *idf*-weighted defaults. Both estimates work well, but experiments with additional test collections will be required to determine if one estimate is better. Using the best default estimate for each query type (i.e., for CACM use a fixed default for the probabilistic and an *idf*-weighted default for the Boolean query) does not improve the performance of the combined queries.

Since a default estimate that is inversely proportional to a term's *idf* did not consistently improve performance, estimates in which the default is directly proportional to *idf* were examined. With this interpretation, the absence of a common term is stronger evidence that the term should not be assigned to the document than the absence of a rare term. These estimates performed significantly worse than either the fixed or the original *idf*-weighted estimates.

While there is little reason to prefer fixed or idf-weighted defaults, the important results here are 1) a non-zero default gives significantly better performance for both collections, and 2) weighted estimates that are inversely proportional to *idf* work better than those that are directly proportional.

8.2.2.4 Combining the *tf* and *idf* estimates

A large number of functions for combining the *tf* and *idf* estimates were tested. These functions have the general form

$$P(t_i|d_j = \text{true}) = \alpha + \beta * tf + \gamma * idf + \delta * tf * idf$$

where $0.4 \leq \alpha \leq 0.6$ and β, γ, and δ were chosen to produce a probability in the range [0..1]. The best performance is generally achieved when $\beta = \gamma = 0.0$ and only the *tf.idf* product term remains. In a final set of experiments, changes to the relative weight of the *tf* and *idf* components of this product term were tested, but changing weights did not significantly improve performance.

Several ranking functions were developed during these experiments that perform well when compared to conventional retrieval models. The performance of many of the best functions is quite similar, but a good overall belief estimate is given by

$$P(t_i|d_j = \text{true}) \quad = \quad 0.4 + 0.6 * tf * idf \tag{8.2}$$

Table 8.9: Comparison of probabilistic and network model performance

	Precision (% change) – 50 queries			
	CACM		CISI	
Recall	Probabilistic	Network	Probabilistic	Network
10	60.2	67.2 (+11.7)	34.8	37.3 (+7.0)
20	48.3	56.2 (+16.4)	26.3	29.1 (+10.6)
30	41.0	47.6 (+16.0)	20.4	21.5 (+5.5)
40	30.9	41.3 (+33.6)	17.0	18.4 (+8.3)
50	26.5	34.4 (+30.2)	15.0	15.9 (+5.8)
60	21.6	29.1 (+34.8)	13.2	13.5 (+2.2)
70	15.0	20.3 (+35.5)	10.7	11.6 (+8.3)
80	11.7	16.3 (+39.1)	9.3	9.2 (−1.7)
90	6.4	11.3 (+76.1)	7.4	7.1 (−4.8)
100	4.4	8.7 (+99.8)	5.5	4.8 (−14.0)
average	26.6	33.3 (+25.0)	16.0	16.8 (+5.3)

$$P(t_i|\text{all parents false}) \quad = \quad 0.4.$$

Variations that work about as well use log normalization for the *tf* component or an *idf*-weighted default (e.g., $P(t_i|\text{all parents false}) = 0.4 - (0.2 * idf)$).

While this estimate works well for both collections and for all query types, it is probably not the best that can be achieved. Our objective in these experiments was to gain a better understanding of the major factors that influence the performance of the retrieval model and how traditional information retrieval weightings could be implemented in the inference network model. Further research can certainly improve these estimates, but it is difficult to estimate how much additional performance can gained.

8.3 Baseline results

Given the strategies for estimating belief at nodes in the network, we now wish to compare the performance of the network model with that of conventional probabilistic (Section 8.3.1) and Boolean (Section 8.3.2) models.

8.3.1 Probabilistic retrieval

It is possible to build inference networks that are equivalent to conventional probabilistic systems if exactly the same indexing strategies are used. In practice, minor variations in the way documents are parsed and indexed will result in small performance differences even when the network form implements a ranking function that is equivalent to that used in the probabilistic system. Using the estimates of the last section, however, it is possible to build networks that perform better than conventional probabilistic models. Table 8.9 shows retrieval performance of the network model compared to a baseline probabilistic model that uses *tf.idf* weighting. Performance improves 25.0% for the CACM collection and 5.3% for CISI. The performance of the network model is better than the probabilistic model at a significance level of 0.001 for CACM and 0.062 for CISI.

These results lead us to accept a strengthened version of Hypothesis 1:

> Given equivalent document representations, query forms, and assumptions about the match between indexing and query vocabularies, the inference network model will improve performance significantly when compared to conventional probabilistic retrieval.

Table 8.10: Comparison of Boolean and network models on CACM

	Precision (% change) – 50 queries		
Recall	Boolean	BL1	BL2
10	46.2	67.6 (+46.3)	66.2 (+43.1)
20	39.3	58.8 (+49.9)	55.5 (+41.5)
30	32.1	50.2 (+56.3)	46.9 (+46.3)
40	26.1	45.2 (+72.8)	41.4 (+58.2)
50	23.6	39.8 (+68.3)	35.4 (+49.8)
60	21.3	33.6 (+57.7)	29.5 (+38.7)
70	13.7	22.0 (+60.0)	22.5 (+63.9)
80	10.5	18.1 (+72.8)	17.8 (+70.3)
90	6.9	11.4 (+65.1)	11.0 (+58.8)
100	5.0	7.9 (+58.9)	8.8 (+75.8)
average	22.5	35.4 (+57.7)	33.5 (+49.1)

The demonstrated performance improvements can be attributed primarily to the use of a default probability. Probabilistic models could be formulated to use the same default strategy and to achieve similar improvements.

8.3.2 Boolean retrieval

Establishing a reasonable baseline for Boolean queries is problematic since conventional Boolean retrieval does not rank documents. In order to simulate Boolean queries, we built inference networks with binary belief estimates for $P(t_i|d_j)$ and used the normal network evaluation procedures. This effectively assigns all documents that satisfy the query a belief of 1.0, all those that do not a belief of 0.0, and breaks ties by sorting on document identifier which places newer documents higher in the ranking. Tables 8.10 and 8.11 compare performance of the network model with conventional Boolean retrieval. Performance improves by 57.7% (BL1) and 49.1% (BL2) for CACM and by 65.3% for CISI. The performance of the network

Table 8.11: Comparison of Boolean and network models on CISI

	Precision (% change) – 35 queries		
Recall	Boolean	Network	
10	23.7	45.3	(+91.3)
20	20.9	32.8	(+57.2)
30	14.6	24.6	(+68.5)
40	13.1	20.5	(+56.2)
50	12.1	17.9	(+47.5)
60	9.2	15.0	(+62.8)
70	6.9	12.7	(+85.0)
80	5.6	9.5	(+69.0)
90	4.9	7.1	(+45.1)
100	3.8	4.3	(+14.2)
average	11.5	19.0	(+65.3)

Table 8.12: Average precision for p-norm and inference network Booleans

	Average Precision – 3 recall points	
	Best p-norm	Network Boolean
CACM	33.1 (p=2)	38.1 (+15.1%)–BL1 35.6 (+7.8%)–BL2
CISI	18.4 (p=1)	19.2 (+4.3%)

model is better than the Boolean model at a significance level of 0.001 for both CACM query sets and 0.001 for CISI.

Our results are compared with those reported for the Extended Boolean or p-norm model of Salton, Fox, and Wu [SFW83] in Table 8.12. For this comparison we use average precision at three recall levels (25%, 50%, and 75%) rather than our customary ten levels in order to permit comparison with the published results. Note that for the CACM collection, Salton, Fox, and Wu include two queries (author searches) that are not used in our experiments.

Table 8.13: Comparison of probabilistic and Boolean searches

	Precision (% change)				
	CACM			CISI	
Recall	prob.	BL1	BL2	prob.	Boolean
10	67.2	67.6 (+0.6)	66.2 (−1.6)	38.3	45.3 (+18.3)
20	56.2	58.8 (+4.7)	55.5 (−1.2)	27.9	32.8 (+17.3)
30	47.6	50.2 (+5.5)	46.9 (−1.3)	20.0	24.6 (+23.3)
40	41.3	45.2 (+9.4)	41.4 (+0.2)	17.5	20.5 (+17.4)
50	34.4	39.8 (+15.5)	35.4 (+2.8)	15.5	17.9 (+15.3)
60	29.1	33.6 (+15.2)	29.5 (+1.4)	12.9	15.0 (+16.1)
70	20.3	22.0 (+8.1)	22.5 (+10.8)	11.2	12.7 (+13.7)
80	16.3	18.1 (+10.7)	17.8 (+9.0)	9.0	9.5 (+4.7)
90	11.3	11.4 (+0.6)	11.0 (−3.3)	7.0	7.1 (+1.7)
100	8.7	7.9 (−9.4)	8.8 (+0.2)	4.7	4.3 (−8.3)
average	33.3	35.4 (+6.6)	33.5 (+0.7)	16.4	19.0 (+15.7)

For both CACM and CISI, the network evaluation of the Boolean queries is better than the best p-norm evaluation. For CACM, average precision for network Booleans is 15.1% better than for the p-norm model for one set of Boolean queries and 7.8% better for the other set. Note that these results should be interpreted cautiously since the details of the precision/recall computation for the published p-norm results are not known.

Salton, Fox and Wu report that the best p-norm interpretation of the Boolean queries outperforms cosine-correlation searches for both CACM and CISI. Our results also show that the Boolean interpretation consistently outperforms the probabilistic (see Table 8.13). This result clearly depends on the quality of the Boolean queries that were generated from the original natural language versions. It would be easy to build Boolean queries that perform poorly and one would expect performance to improve if domain knowledge was used to expand the set of query terms. The Boolean queries in the test set, however, do not add new terms or domain knowledge. Performance improvements appear to arise because the Boolean queries

capture structural information in the queries (phrase structure, compound nominals, and negation) that is not exploited in probabilistic queries. The potential for encoding linguistic structure in Boolean or Boolean-like expressions is an important area for future research.

8.3.3 Phrase-structured Booleans

Since the Booleans appear to capture linguistic structure in the original natural language query, a new set of Boolean queries was built using data collected in an earlier experiment in which important word pairs in the CACM queries were manually identified. The individual terms in these pairs were combined using *and* and the pairs were combined using a Boolean *or* in one set of experiments and a probabilistic *sum* operator in a second. These new queries represent an attempt to retrieve documents containing the significant phrases in the original queries rather than the individual terms. The documents retrieved by these queries are a subset of those retrieved by the original natural language queries since no new terms are introduced.

The performance of these word-pair queries when compared to the original natural language queries is shown in Table 8.14. With both the *or* and *sum* combinations, performance is significantly worse. Performance of the word-pair queries is also significantly worse than the original Boolean queries (Table 8.15). It is clear that the manually constructed Boolean queries are capturing important linguistic information that the word-pair queries are not.

When the word-pair queries are combined with the natural language version (Table 8.16), the *or* combination degrades performance slightly while the *sum* combination improves performance somewhat (significance level of 0.344). In both cases, performance at high recall levels is poor, while performance at high precision

Table 8.14: Word pair versus NL queries

	Precision (% change) – 50 queries		
Recall	NL	or	sum
10	67.2	63.9 (−5.0)	64.9 (−3.4)
20	56.2	54.6 (−2.8)	56.0 (−0.4)
30	47.6	47.8 (+0.6)	48.2 (+1.3)
40	41.3	38.0 (−7.9)	38.5 (−6.7)
50	34.4	32.9 (−4.4)	33.3 (−3.2)
60	29.1	25.7 (−11.9)	26.1 (−10.3)
70	20.3	17.5 (−13.7)	17.5 (−13.8)
80	16.3	13.6 (−17.0)	13.4 (−18.1)
90	11.3	8.4 (−25.6)	8.4 (−26.1)
100	8.7	5.8 (−33.1)	5.8 (−33.1)
average	33.3	30.8 (−7.3)	31.2 (−6.1)

Table 8.15: Word pair versus Boolean queries

	Precision (% change) – 50 queries		
Recall	Boolean	or	sum
10	67.6	63.9 (−5.5)	64.9 (−4.0)
20	58.8	54.6 (−7.2)	56.0 (−4.9)
30	50.2	47.8 (−4.6)	48.2 (−3.9)
40	45.2	38.0 (−15.8)	38.5 (−14.7)
50	39.8	32.9 (−17.2)	33.3 (−16.2)
60	33.6	25.7 (−23.5)	26.1 (−22.1)
70	22.0	17.5 (−20.2)	17.5 (−20.3)
80	18.1	13.6 (−25.0)	13.4 (−26.0)
90	11.4	8.4 (−26.0)	8.4 (−26.5)
100	7.9	5.8 (−26.1)	5.8 (−26.1)
average	35.4	30.8 (−13.0)	31.2 (−11.9)

Table 8.16: Word-pair combined with NL queries

	Precision (% change) – 50 queries		
Recall	NL	or	sum
10	67.2	66.9 (−0.5)	68.9 (+2.4)
20	56.2	57.4 (+2.1)	59.4 (+5.8)
30	47.6	51.3 (+7.8)	52.4 (+10.2)
40	41.3	41.6 (+0.9)	43.9 (+6.3)
50	34.4	35.4 (+2.9)	36.6 (+6.2)
60	29.1	27.9 (−4.2)	28.1 (−3.4)
70	20.3	19.4 (−4.6)	19.9 (−2.0)
80	16.3	15.0 (−8.3)	16.3 (−0.0)
90	11.3	9.4 (−17.4)	10.1 (−11.2)
100	8.7	6.8 (−21.7)	7.3 (−16.4)
average	33.3	33.1 (−0.4)	34.3 (+3.1)

levels is relatively good, especially for the *sum* combination. Part of the reason for the performance differential stems from the fact that the word-pair queries rank highly only those documents that have at least two terms in common with the query. As a result, they tend to rank poorly at the high recall levels where documents typically have only one or two terms in common with the query. These results suggest that it may be possible to improve performance by modifying scores only for documents that are highly ranked by word-pair queries.

The queries used in Tables 8.14 through 8.16 contain only word pairs. Since the phrase model discussed in Chapter 7 suggests that the queries should also contain significant single terms from the original queries a new set of queries were created that contained these single terms plus content-bearing terms that occurred in the word pairs. In order to reduce the chance for bias, the choice of content-bearing terms was not made by the author.

These augmented queries were intended to simulate the performance that could be achieved if a user marked important phrases (word pairs) and terms in the orig-

Table 8.17: Word-pair versus phrase-structured Booleans

	Precision (% change) – 50 queries					
	or			*sum*		
Recall	word-pair	phrase-structured		word-pair	phrase-structured	
10	63.9	68.8	(+7.7)	64.9	70.9	(+9.1)
20	54.6	60.9	(+11.6)	56.0	62.7	(+12.0)
30	47.8	49.4	(+3.2)	48.2	52.4	(+8.6)
40	38.0	40.6	(+6.6)	38.5	43.1	(+11.8)
50	32.9	34.5	(+4.8)	33.3	35.4	(+6.1)
60	25.7	29.2	(+13.9)	26.1	29.1	(+11.4)
70	17.5	21.0	(+19.6)	17.5	22.4	(+27.9)
80	13.6	17.5	(+29.0)	13.4	18.5	(+38.1)
90	8.4	12.2	(+44.1)	8.4	12.9	(+54.2)
100	5.8	8.4	(+43.3)	5.8	9.3	(+59.9)
average	30.8	34.2	(+11.0)	31.2	35.7	(+14.2)

Table 8.18: Phrase-structured versus NL and Boolean queries

	Precision (% change) – 50 queries							
Recall	NL	BL1		BL2		phrase-structured	
10	67.2	67.6	(+0.6)	66.2	(−1.6)	70.9	(+5.4)
20	56.2	58.8	(+4.7)	55.5	(−1.2)	62.7	(+11.6)
30	47.6	50.2	(+5.5)	46.9	(−1.3)	52.4	(+10.1)
40	41.3	45.2	(+9.4)	41.4	(+0.2)	43.1	(+4.4)
50	34.4	39.8	(+15.5)	35.4	(+2.8)	35.4	(+2.7)
60	29.1	33.6	(+15.2)	29.5	(+1.4)	29.1	(−0.1)
70	20.3	22.0	(+8.1)	22.5	(+10.8)	22.4	(+10.3)
80	16.3	18.1	(+10.7)	17.8	(+9.0)	18.5	(+13.1)
90	11.3	11.4	(+0.6)	11.0	(−3.3)	12.9	(+14.0)
100	8.7	7.9	(−9.4)	8.8	(+0.2)	9.3	(+7.0)
average	33.3	35.4	(+6.6)	33.5	(+0.7)	35.7	(+7.2)

inal queries and this information was used to automatically build a Boolean query that incorporates phrase structure. Table 8.17 shows that these phrase-structured Boolean queries perform significantly better than the original word-pair Booleans and, in fact, perform significantly better than the original natural language queries (Table 8.18) and better than the Boolean queries (Table 8.18 uses the *sum* operator to combine phrases and terms). However, as will be discussed in the next section, when the phrase-structured queries are combined with the natural language queries they do not improve performance as much as do the Boolean queries provided with the test set.

These results are encouraging since they show

1. that phrase information can be used to significantly improve performance of a natural language query, and

2. that it should be possible to automatically generate a Boolean query representation based on information about phrase structure and term importance provided by the user.

8.4 Multiple query representations

In order to test Hypothesis 3:

> The use of multiple query formulations and search strategies will significantly improve retrieval performance when compared to baseline probabilistic searches,

inference networks were built to evaluate both probabilistic and Boolean versions of the query and to combine the results using a weighted-sum matrix. Table 8.19 shows the effect of combining probabilistic and Boolean queries with CACM and

Table 8.19: Performance of combined queries on CACM

	Precision (% change from probabilistic)				
Recall	Probabilistic	BL1	Combined	BL2	Combined
10	67.2	67.6	76.2 (+13.3)	66.2	71.9 (+6.9)
20	56.2	58.8	66.3 (+17.9)	55.5	60.0 (+6.8)
30	47.6	50.2	56.3 (+18.3)	46.9	51.8 (+8.9)
40	41.3	45.2	50.9 (+23.4)	41.4	44.3 (+7.3)
50	34.4	39.8	45.4 (+31.6)	35.4	38.7 (+12.4)
60	29.1	33.6	38.8 (+33.1)	29.5	32.4 (+11.1)
70	20.3	22.0	25.1 (+23.5)	22.5	23.6 (+16.0)
80	16.3	18.1	20.4 (+24.8)	17.8	18.9 (+16.0)
90	11.3	11.4	12.4 (+9.1)	11.0	11.7 (+3.7)
100	8.7	7.9	9.2 (+4.8)	8.8	9.2 (+5.5)
average	33.3	35.4	40.1 (+20.5)	33.5	36.3 (+9.0)

Table 8.20: Performance of combined queries on CISI

	Precision (% change) – 35 queries		
Recall	Probabilistic	Boolean	Combined
10	38.3	45.3 (+18.3)	44.7 (+16.7)
20	27.9	32.8 (+17.3)	36.1 (+29.2)
30	20.0	24.6 (+23.3)	25.0 (+25.0)
40	17.5	20.5 (+17.4)	20.3 (+16.2)
50	15.5	17.9 (+15.3)	18.0 (+16.2)
60	12.9	15.0 (+16.1)	15.2 (+17.1)
70	11.2	12.7 (+13.7)	12.8 (+14.3)
80	9.0	9.5 (+4.7)	9.5 (+5.3)
90	7.0	7.1 (+1.7)	7.0 (+0.7)
100	4.7	4.3 (−8.3)	4.6 (−2.5)
average	16.4	19.0 (+15.7)	19.3 (+17.8)

Table 8.20 shows performance for CISI. Combining queries increased performance by 20.5% (BL1) and 9.0% (BL2) for the CACM collection and by 17.8% for CISI. The combined performance is better than for probabilistic queries at significance levels of 0.002 for both CACM query sets and 0.004 for CISI. The combined performance is better than for Boolean queries at significance levels of 0.001 and 0.002 for CACM but only 0.250 for CISI. These results lead us to accept Hypothesis 3. The performance improvement due to combining queries is one of the most consistent improvements observed in this research; for any reasonable document network, combining probabilistic and Boolean queries gives significant performance improvements.

We originally thought that at least part of the performance improvements arose because the two query types were retrieving different relevant documents so that the combined set contained more relevant documents than retrieved by the separate queries. This is not, however, the case. The documents retrieved by the Boolean queries are a subset of those retrieved by the corresponding probabilistic query. The individuals preparing the Boolean queries rarely added new terms to the query (for CACM, 4 out of 50 queries in each Boolean set contained new terms) and these new terms retrieved no new relevant documents. It appears that the objective in creating the Boolean queries was to capture the structural information present in the natural language versions, not to produce the best possible Boolean queries. If trained searchers were asked to produce high-recall Boolean queries from the natural language descriptions, they would generally use their knowledge of the subject domain and indexing practice to expand the set of terms to include synonyms and related terms. It is likely that these enhanced searches would retrieve relevant documents not found by the probabilistic query and that these queries would perform better than those provided with the test collection.

Table 8.21: NL plus original and phrase-structured Booleans

	Precision (% change) – 50 queries			
Recall	NL	BL1	NL	phrase-structure
10	67.2	76.2 (+13.3)	67.2	73.0 (+8.6)
20	56.2	66.3 (+17.9)	56.2	61.9 (+10.1)
30	47.6	56.3 (+18.3)	47.6	54.5 (+14.5)
40	41.3	50.9 (+23.4)	41.3	46.6 (+12.9)
50	34.4	45.4 (+31.6)	34.4	39.0 (+13.3)
60	29.1	38.8 (+33.1)	29.1	32.1 (+10.2)
70	20.3	25.1 (+23.5)	20.3	23.2 (+14.2)
80	16.3	20.4 (+24.8)	16.3	18.9 (+15.6)
90	11.3	12.4 (+9.1)	11.3	13.2 (+16.6)
100	8.7	9.2 (+4.8)	8.7	9.8 (+12.2)
average	33.3	40.1 (+20.5)	33.3	37.2 (+11.9)

The observed performance improvement, then, is due entirely to the fact that the normalized sum of the beliefs produces a better ranking than the rankings produced by the probabilistic or Boolean queries alone.

When first analyzing these results we were somewhat surprised to find that a set of belief estimates that improves the performance of both the probabilistic and Boolean queries need not improve their combined performance. To see why this is possible, assume that we have rankings for Boolean and probabilistic queries, the probabilistic ranking is very good, the Boolean ranking is weak, and that the probabilities in the probabilistic ranking are large when compared to the Boolean ranking. When the rankings are combined, the probabilistic beliefs will dominate the combined ranking and produce a good result. If we now adopt an estimate that produces similar (or slightly improved) rankings for both queries, but increases the beliefs in the weak Boolean ranking while reducing the beliefs in the probabilistic ranking, the combined ranking will be degraded.

Experiments reported in the last section showed that phrase-structured Booleans performed better than the Boolean queries supplied with the CACM collection. When the two Boolean forms are combined with the natural language query, however, the phrase-structured Booleans do not improve performance as much as the original Booleans (Table 8.21). It appears that the evidence provided by the phrase-structured queries is similar to that provided by the natural language queries; the original Booleans are capturing additional linguistic information.

To determine the degree to which the natural language and Boolean rankings agreed for the CACM test queries, correlation coefficients (Pearson product-moment) were computed for the rankings produced by each query. The rankings agreed reasonably well (no negative coefficients), ranging from 0.286 to 1.0 for BL1 and from 0.282 to 1.0 for BL2 with mean rank coefficients of 0.731 and 0.735, respectively (using the belief estimates of equation 8.2). Unfortunately, the correlation coefficient does not appear to be a good predictor of the performance of the combined ranking; identical rankings can both be wrong, and dissimilar rankings can be combined to produce a good result.

Attempts to weight the query types either by scaling beliefs to a similar range or assigning fixed weights to the query types did not improve performance. It is likely, however, that information about the relative quality of the queries (e.g., user judgements) could be used to weight the contribution of each query and to improve performance. In the next section we will present results which show that query weighting can be useful when adding evidence that is known to be weak.

The actual belief values produced by the two query types are quite variable and it is difficult to predict whether one ranking will dominate the combined result. Probabilistic beliefs tend to be more uniformly distributed between a minimum that is either fixed (fixed default belief) or determined by the number of query terms

(*idf*-weighted default) and a maximum determined by the number of terms and the term weights. Boolean belief values depend heavily on the structure of the Boolean expression. In practice, *and*-structured queries generally rank a relatively small number of documents highly after which belief values drop off rapidly. *Or*-structured queries tend to produce a more uniform distribution of beliefs.

Croft and Thompson [CT84] found that different query representations or strategies worked better for some queries than others and that it was difficult to predict which strategy would work best with a given query. In their work they tried to select a query evaluation strategy based on different query features (query length, sum and average of *idf* weights), but found that these features did not predict which strategy would work well. One of the strengths of the network model is that it is not necessary to predict which query representation will perform well. Given reasonable query representations, the combined performance is better than that achieved by either representation separately.

8.5 Multiple document representations

In addition to the normal bibliographic fields, the CACM collection includes author assigned Computing Reviews (CR) categories for roughly half of the articles (CR categories were not used before 1968). The set of Computing Review categories used evolved during the period covered by the collection, but the changes affected a relatively small number of categories (see [Lew89] for detailed description of the use of Computing Reviews categories in CACM). The CR categories represent an additional document representation that can be used to test

Table 8.22: Performance of CR category searches

	Precision – 50 queries	
Recall	NL	Boolean
10	10.9	10.9
20	7.3	7.4
30	5.0	5.2
40	4.1	4.2
50	3.6	3.6
60	2.6	2.6
70	1.6	1.6
80	1.4	1.4
90	1.1	1.1
100	0.9	0.9
average	3.9	3.9

Hypothesis 2 – The use of multiple document representations will significantly improve retrieval performance when compared to equivalent networks without the additional representations.

The CACM queries do not contain CR categories. As an initial test, the author manually assigned Computing Reviews categories to each query. This approach represents a reasonable upper bound for performance with CR categories since it is unlikely that a significantly better assignment could be made automatically. These categories were then used in two experiments: one in which the CR categories were treated as terms in a natural language query (i.e., given normal *tf.idf* weights and combined using weighted-sum) and a second in which the CR categories were treated as terms in a Boolean *or* query. The results for both experiments, shown in Table 8.22, are poor. CR categories, by themselves, are not an effective retrieval tool for the CACM collection.

Although CR categories do not work well by themselves, they may still represent an additional form of evidence that can improve the performance of a natural

Table 8.23: Performance of CR categories added to natural language query

	Precision (% change) – 50 queries		
Recall	NL	new terms	sep. query
10	67.2	66.4 (−1.3)	39.2 (−41.7)
20	56.2	53.3 (−5.1)	30.9 (−44.9)
30	47.6	47.7 (+0.3)	27.3 (−42.7)
40	41.3	38.5 (−6.8)	21.8 (−47.3)
50	34.4	33.4 (−3.0)	18.5 (−46.4)
60	29.1	27.1 (−7.0)	15.5 (−46.8)
70	20.3	20.0 (−1.8)	12.8 (−37.1)
80	16.3	16.1 (−1.6)	10.0 (−38.9)
90	11.3	11.1 (−2.0)	6.4 (−43.2)
100	8.7	8.6 (−2.1)	3.9 (−55.7)
average	33.3	32.2 (−3.2)	18.6 (−44.0)

language query. Table 8.23 shows performance when CR categories are added as terms to the original query and when they are added as a separate natural language query.

The addition of the CR categories as new terms to the natural language query degrades performance slightly. Adding CR categories as a separate Boolean query significantly degrades performance. Since the CR categories were assigned to queries by a domain expert, these results are not encouraging.

Since earlier work by Fox [Fox83b] suggested that CR categories could improve performance when treated as relatively weak evidence when compared to the natural language query, we ran a series of experiments in which the CR categories were added as a separate query with reduced weight. The best performance was achieved when the CR categories were scaled by a factor of 0.15 (Table 8.24). Reducing the evidential weight of the CR categories does improve their performance significantly, but when combined with the natural language query, they improve performance only slightly.

Table 8.24: Effect of reducing the weight of CR categories

	Precision (% change) – 50 queries		
Recall	NL	unweighted	weighted
10	67.2	39.2 (−41.7)	69.7 (+3.7)
20	56.2	30.9 (−44.9)	56.0 (−0.3)
30	47.6	27.3 (−42.7)	48.5 (+2.0)
40	41.3	21.8 (−47.3)	40.7 (−1.5)
50	34.4	18.5 (−46.4)	34.8 (+0.9)
60	29.1	15.5 (−46.8)	29.4 (+0.8)
70	20.3	12.8 (−37.1)	21.5 (+5.7)
80	16.3	10.0 (−38.9)	17.1 (+4.5)
90	11.3	6.4 (−43.2)	11.5 (+1.6)
100	8.7	3.9 (−55.7)	8.8 (+0.7)
average	33.3	18.6 (−44.0)	33.8 (+1.6)

To check these results, a second set of CR category assignments was done by another researcher who was quite familiar with the CR categories and their use in the CACM collection. For this assignment, two sets of CR categories were assigned to each query, a primary set which described the main subject area of the query, and a secondary set which included other related CR categories. This was a carefully prepared set with an average of 6.7 CR categories assigned to each query (2.3 of which were primary); the original assignment contained only 2.4 CR categories per query. Performance when this set is added as new query terms is shown in Table 8.25.

When the full set of assigned CR categories is used, performance degrades with the worst performance loss at high precision levels. Using only the primary categories, however, improves performance (significance level of 0.008). When the CR categories are added as a separate scaled query (Table 8.26) performance improves further (significance level of 0.008).

Table 8.25: Expert-assigned CR categories as new terms

	Precision (% change) – 50 queries		
Recall	NL	all CR cat.	primary cat.
10	67.2	62.4 (−7.2)	65.1 (−3.2)
20	56.2	52.7 (−6.2)	55.8 (−0.8)
30	47.6	46.4 (−2.5)	50.0 (+5.1)
40	41.3	39.2 (−5.1)	43.4 (+5.1)
50	34.4	34.7 (+0.7)	38.2 (+11.0)
60	29.1	26.9 (−7.5)	30.0 (+3.1)
70	20.3	20.7 (+1.7)	22.0 (+8.3)
80	16.3	16.6 (+1.5)	17.5 (+7.2)
90	11.3	11.9 (+5.4)	13.3 (+17.0)
100	8.7	8.8 (+1.1)	10.0 (+14.2)
average	33.3	32.0 (−3.7)	34.5 (+3.8)

A recent study by Crouch, Crouch, and Nareddy [CCN90] found that CR categories could be used to improve performance on the CACM collection by assigning to the query those categories assigned to documents that were ranked highly by the natural language query. The essential idea is to use the initial query to identify a set of potentially relevant documents and to use the CR categories assigned to these documents as an additional query representation. This approach has the significant advantage that CR categories can be assigned automatically. To simulate their approach we evaluated each query, then added the CR categories assigned to the top ten documents as a separate query (scaled by 0.15).

Table 8.27 shows results when all CR categories from the top ten documents are added to the natural language query and when only those CR categories assigned to more than one of the top ten documents are added. Using all CR categories assigned to the top ten documents results in an average of 18.0 CR categories per query. Using only those CR categories assigned to more than one document, 4.7 CR categories are assigned to each query. In both cases, the automatically assigned CR

Table 8.26: Expert-assigned CR categories as separate query

	Precision (% change) – 50 queries		
Recall	NL	all CR cat.	primary cat.
10	67.2	70.1 (+4.2)	71.1 (+5.7)
20	56.2	57.9 (+2.9)	59.4 (+5.8)
30	47.6	50.2 (+5.6)	52.5 (+10.3)
40	41.3	42.8 (+3.6)	44.3 (+7.3)
50	34.4	37.2 (+7.8)	38.4 (+11.5)
60	29.1	29.4 (+0.9)	29.3 (+0.7)
70	20.3	21.6 (+6.0)	22.1 (+8.9)
80	16.3	17.6 (+7.6)	17.1 (+4.8)
90	11.3	12.4 (+9.1)	12.3 (+8.2)
100	8.7	9.4 (+7.5)	9.0 (+3.6)
average	33.3	34.8 (+4.7)	35.6 (+6.9)

categories perform about as well as the best manual assignment, but when added to the natural language query no significant improvement is achieved (this result is sensitive to the weighting scheme used in the network – for some weightings it is possible to show significant improvement).

Since the Crouch study found a very significant improvement using CR categories, our results do not support theirs. We note, however, that our baseline precision is 8% better than their best result after adding CR categories; it may be possible to achieve greater improvement when starting from a weaker baseline.

There are several potential explanations for the mixed performance of the CR categories. First, they are only assigned to half of the CACM collection, so that any relevant documents published before 1968 can only be ranked lower by the inclusion of CR categories in the query. Further, since documents in the second half of the collection are judged relevant more often (83% of all relevance judgements are for documents in the second half of the collection), a randomly selected document with a CR category assignment is more than four times as likely to be judged relevant as

Table 8.27: Using feedback to assign CR categories

	Precision (% change) – 50 queries			
Recall	NL	primary cat.	all tops	frequent tops
10	67.2	71.1 (+5.7)	66.0 (−1.9)	65.8 (−2.2)
20	56.2	59.4 (+5.8)	54.9 (−2.3)	53.9 (−4.2)
30	47.6	52.5 (+10.3)	47.0 (−1.2)	46.4 (−2.5)
40	41.3	44.3 (+7.3)	41.2 (−0.1)	42.0 (+1.7)
50	34.4	38.4 (+11.5)	35.7 (+3.6)	35.3 (+2.5)
60	29.1	29.3 (+0.7)	29.7 (+1.9)	30.7 (+5.5)
70	20.3	22.1 (+8.9)	20.7 (+2.1)	21.8 (+7.1)
80	16.3	17.1 (+4.8)	16.9 (+3.7)	17.6 (+7.7)
90	11.3	12.3 (+8.2)	11.4 (+1.1)	11.6 (+2.8)
100	8.7	9.0 (+3.6)	9.0 (+2.9)	8.7 (−0.1)
average	33.3	35.6 (+6.9)	33.3 (+0.0)	33.4 (+0.4)

a randomly selected document with no CR category assignment. As a result, one would expect performance to improve if we raised belief in all documents containing a CR category assignment without regard to query or document content.

Second, the assignment of CR categories to both documents and queries is subjective and error-prone. Finally, the CR categories do not represent many of the queries in the test set well. Both researchers who assigned CR categories to queries had difficulty because many of the queries dealt with specializations that had not yet been incorporated in the Computing Reviews structure. For example, many of the queries dealt with specialized aspects of distributed systems. Unfortunately, the CR classification had no appropriate code for distributed systems and queries had to be assigned to loosely related categories (e.g., category 4.3 – Supervisory Systems). Newer versions of the CR classification provide a richer set of categories that would better match the queries in the test collection, but new categories are not retrospectively assigned to documents in the collection. This mismatch is a problem for controlled vocabularies in any rapidly growing field.

These experimental results do not support any firm conclusions regarding Hypothesis 2. Significant improvement is possible and the poor performance observed here is largely due to problems with using the CR categories as descriptors with this collection. However, significant improvement appears to require assignment by a human expert so the question of whether multiple document representations can improve performance in an operational setting remains open. Answering this question will require test collections with richer document representations or natural language techniques that exhibit a better understanding of the query.

8.6 Extended model

Two features of the extended model, citation links and nearest neighbor clustering, were used to test Hypothesis 4:

> The incorporation of dependencies between documents (citations and nearest neighbor links) will significantly improve retrieval performance when compared to equivalent networks without the additional dependencies.

These types of evidence were found to be useful in spreading activation research. For both types of evidence, we used data developed for spreading activation experiments with I^3R [TC89].

Essentially, we treat the existence of a citation or nearest neighbor link as evidence that the cited or clustered document has content that is similar to the original. In Figure 8.1 document d_1 has been instantiated. All remaining documents in the network are set to false, except those that are related to d_1 by citation or nearest neighbor links. The belief accorded these related documents is based on the type and strength of the relationship with d_1. The effect of this evidence is to raise

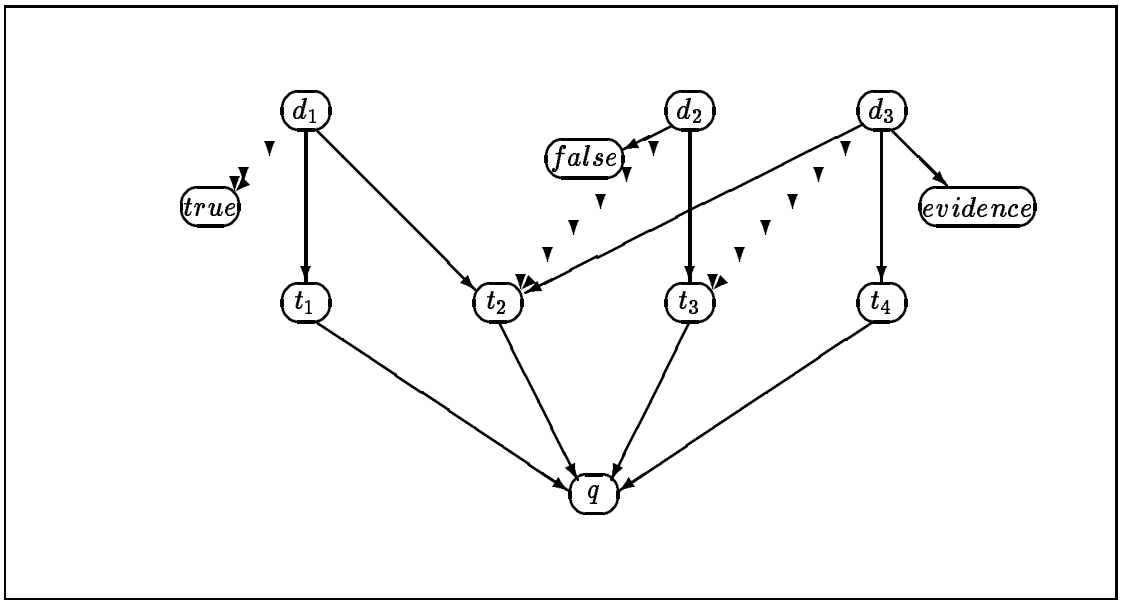

Figure 8.1: Evidence in the extended model

the belief in all terms contained in related documents which effectively adds terms to d_1's representation and reinforces belief in terms held in common. In the example, the presence of a link from d_1 to d_3 adds terms t_3 and t_4 to d_1's representation, strengthens belief in t_2 given d_1, and leaves belief in t_1 unchanged.

There are, then, two cases for which we must provide revised belief estimates: a) when one or more related documents contain a term in common with the original, and b) when new terms are introduced. Starting with the belief estimates developed for the basic model we are interested in adjusting our estimate for

$$P(t_i|d_j = \text{true}) = \alpha + (1 - \alpha) * tf * idf \tag{8.3}$$

based on this new evidence. Note that our estimate for the default belief is unaffected. Since we still wish to base the maximum belief that can be achieved for a term on that term's *idf* value, we are looking for a new estimate (call it ϵ) for the *tf* component in equation 8.3. In case a) we start with a *tf* estimate for the original

document and must revise it based on the strength of the link or on the number of times the term occurs in related documents. In case b) we must produce an entirely new estimate since the term did not occur in the original document.

When our estimate for the *tf* component is interpreted as a normalized term frequency, it takes on values in the range $0 < \epsilon < 1$; this is the appropriate interpretation for case a). In case b), however, it is useful to allow negative values for ϵ. For example, using an estimate with an *idf*-weighted default like

$$\begin{aligned} P(t_i|d_j = \text{true}) &= 0.4 + 0.6 * \epsilon * nidf \\ P(t_i|d_j = \text{false}) &= 0.4 - (0.2 * nidf) \end{aligned} \tag{8.4}$$

we can choose $\epsilon = -1/3$ for new terms to produce a belief estimate that is the same as the default. In this case terms added through related documents have no effect on our belief in the query and performance is identical to that for a network without the additional evidence. For small ϵ we are asserting that new terms are probably not correct descriptors for the original document. Useful estimates for new-term ϵ lie between the value that produces the default belief and 1.0.

8.6.1 Citations

Each document d in the CACM collection contains a list of documents that d cites and a list of documents which cite d (assuming that the other document is in the test collection). A citation link is created for each document that d cites and we say that this link belongs to d or that d has a citation link.

Summary statistics for citations in the CACM collection are shown in Table 8.28. There are 6063 citations in the collection for an average of 1.9 citations per document. 1747 or 54.5% of the documents have one or more citations. The original collection contains 79,243 unique documents/term pairs (postings); 25.9% of these

Table 8.28: Use of citations and nearest neighbor clusters in CACM

	citations	nearest neighbors
Terms in basic collection	5493	5493
Postings in basic collection	79243	79243
Terms to which new postings added	4379 (80%)	2499 (46%)
Postings modified	20491	12347
Postings added	128154	13145
Postings in extended collection	207397	92388
Number of citation/nn links in collection	6063	3317
Number of documents with citation/nn links	1747	1928

are modified as a result of citations. Citations add 128,154 new postings to the collection (162% of the original) and account for 61.8% of all postings. If we count both new and modified postings, citations affect belief values for 71.2% of all postings.

Previous work with citations suggests that they represent a good source of evidence about document content [SFV83, BM85, PW89], but most of this research assumes that we start with a set of documents that are known to be relevant and use citations to find and rank additional documents. Previous work offers little guidance about the relative importance of terms in cited documents. Under the assumption that the contributions of new and common terms are relatively independent, we first found a reasonable estimate for new term belief while leaving the belief values for terms held in common with the original document unchanged. We then varied beliefs for the common terms to find a good overall estimate.

The terms added by citation links do not appear to be very strong sources of evidence. The number of terms added to a document is large, generally the number of terms added exceeds the number of original terms, in some cases, by a factor of 25 or more. Many of these terms do not appear to be good descriptors for the original document.

Table 8.29: Effect of varying ϵ for new terms

	Precision (% change) – 50 queries				
Recall	$\epsilon = -1/3$	$\epsilon = -0.2$	$\epsilon = -0.15$	$\epsilon = -0.1$	$\epsilon = 0.0$
10	76.2	77.3 (+1.6)	77.6 (+1.9)	78.1 (+2.5)	77.8 (+2.2)
20	66.3	68.4 (+3.1)	69.3 (+4.6)	69.9 (+5.5)	69.9 (+5.5)
30	56.3	58.5 (+4.0)	59.2 (+5.1)	58.4 (+3.8)	57.6 (+2.4)
40	50.9	52.4 (+2.9)	51.7 (+1.5)	50.5 (−0.8)	51.0 (+0.1)
50	45.4	47.7 (+5.3)	47.2 (+4.0)	46.4 (+2.3)	45.8 (+1.0)
60	38.8	40.9 (+5.5)	40.7 (+5.0)	39.6 (+2.2)	39.0 (+0.7)
70	25.1	27.7 (+10.5)	27.9 (+11.0)	27.7 (+10.4)	27.1 (+8.2)
80	20.4	21.6 (+6.1)	22.0 (+8.2)	22.3 (+9.7)	22.5 (+10.6)
90	12.4	13.3 (+7.4)	13.5 (+9.2)	13.9 (+12.6)	14.2 (+15.1)
100	9.3	9.7 (+4.6)	9.7 (+5.0)	9.6 (+3.2)	9.6 (+3.2)
average	40.1	41.8 (+4.2)	41.9 (+4.5)	41.6 (+3.9)	41.5 (+3.4)

Table 8.29 shows retrieval performance for values of ϵ (equation 8.4) ranging from -0.33 to 0.0 for combined queries. Performance is relatively flat across this range and drops off above 0.0. As expected, these terms appear to represent relatively weak evidence. They are less important than any term occurring in the original document and provide only slightly stronger evidence than the absence of the same term ($\epsilon = -0.33$). Attempts to use the term's *idf* score when estimating ϵ did not improve performance over the fixed estimate.

Adopting $\epsilon = -0.15$ as the baseline for new terms, we then examined estimates for terms held in common. Several estimates based on the frequency of term occurrence in cited documents and on the term's *idf* score were tried, but the simple strategy of increasing the belief estimate by 10% of the difference between its original value and the maximum achievable for the term worked as well as more complicated estimates. Again, this suggests that the fact that a term occurs in a cited document is relatively weak additional evidence that it should be assigned to the original.

Table 8.30: Performance improvement due to citations

	Precision – 50 queries	
Recall	baseline	with citations
10	67.2	69.6 (+3.5)
20	56.2	59.5 (+5.9)
30	47.6	49.8 (+4.8)
40	41.3	44.7 (+8.2)
50	34.4	39.0 (+13.3)
60	29.1	31.8 (+9.2)
70	20.3	21.9 (+7.9)
80	16.3	18.5 (+13.4)
90	11.3	12.4 (+9.7)
100	8.7	9.4 (+7.1)
average	33.3	35.7 (+7.3)

Using $\epsilon = -0.15$ for new terms and increasing belief by 10% of the available gain for terms held in common, citation information does improve performance. Table 8.30 compares performance for natural language queries. Performance with citation information is better than the baseline natural language query at a significance level of 0.004. Performance also improves for Boolean and combined queries, but the significance levels are weaker (0.063 and 0.031 respectively).

Although the use of citation information does improve performance, citations are relatively expensive to implement. Citations more that double the number of postings for a collection which more than doubles the size of the inverted file and substantially increases the average length of a posting list. For large collections, storage and processing costs may make the use of citations unattractive.

8.6.2 Nearest neighbor clusters

The use of nearest neighbor clusters is quite similar to the use of citation links. They differ mainly in that nearest neighbor clusters have a measure of document similarity

and that the size of the nearest neighbor clusters (typically 1 to 3 documents) is smaller than for citations. Given the similarity of citation and nearest neighbor links, the same estimation procedures can be applied.

Summary statistics for nearest neighbor clusters are shown in Table 8.28. There are 3317 nearest neighbor links in the collection for an average of 1.0 links per document. 1928 or 60.2% of the documents have one or more nearest neighbor links. The original collection contains 79,243 postings, 15.6% of which are modified as a result of a nearest neighbor link. Nearest neighbor links add 13,145 new postings to the collection (16.6% of the original). These new postings account for 14.2% of all postings in the expanded collection. If we count both new and modified postings, nearest neighbors affect belief values for 27.6% of all postings. Compared with citations, nearest neighbors add about 1/10 as many postings and affect belief in about half as many of the original terms.

The terms added by nearest neighbor links appear to be much better descriptors for the original documents than those added by citations. When examining the nearest neighbor clusters, we found some odd clusters in which the cluster members were very short records containing text that was nearly identical. For example, document 802 has documents 803, 926, and 1125 as nearest neighbors. All of these are very short records (title, author and date) and all have identical titles ("SYMINV (Algorithm 150")). Documents 802 and 803 have identical titles and dates and differ only in author. It is not known if clusters made up of only very short documents are common. We examined only five clusters, two of which were of questionable value.

Using the same strategy as for citations, we held belief for terms in common fixed and varied ϵ for new terms. Table 8.31 shows performance for values of ϵ for combined queries (probabilistic and Boolean queries behave differently for nearest

Table 8.31: Nearest neighbor performance for new terms– combined queries

	Precision (% change) – 50 queries				
Recall	$\epsilon = -1/3$	$\epsilon = -0.2$	$\epsilon = -0.1$	$\epsilon = 0.0$	$\epsilon = 0.1$
10	76.2	76.5 (+0.4)	76.6 (+0.5)	76.6 (+0.5)	77.5 (+1.8)
20	66.3	66.3 (+0.0)	66.5 (+0.3)	66.8 (+0.7)	66.8 (+0.8)
30	56.3	56.2 (−0.1)	56.2 (−0.0)	56.0 (−0.4)	56.1 (−0.3)
40	50.9	50.8 (−0.2)	50.9 (+0.0)	50.8 (−0.2)	50.8 (−0.2)
50	45.3	46.0 (+1.5)	46.1 (+1.7)	46.0 (+1.4)	45.9 (+1.3)
60	38.8	39.1 (+1.0)	39.1 (+1.0)	39.1 (+0.8)	39.1 (+0.8)
70	25.1	25.2 (+0.6)	25.3 (+0.8)	25.4 (+1.2)	25.4 (+1.4)
80	20.4	20.9 (+2.4)	21.0 (+3.2)	21.0 (+3.2)	20.9 (+2.5)
90	12.4	12.5 (+1.0)	12.5 (+0.9)	12.4 (+0.5)	12.4 (+0.3)
100	9.2	9.2 (+0.1)	9.1 (−0.7)	9.1 (−0.4)	9.0 (−1.2)
average	40.1	40.3 (+0.5)	40.3 (+0.7)	40.3 (+0.6)	40.4 (+0.8)

neighbors. The performance of probabilistic queries increases slightly for increasing ϵ while performance for Booleans decreases). Combined performance improves slightly for $\epsilon > -1/3$ and changes little across the range of interest. The slight increase is not significant.

Since nearest neighbor links include a weight describing the degree of similarity between documents, estimates for ϵ based on the nearest neighbor weights were tried, but these estimates did not significantly improve performance over the fixed estimates.

Using $\epsilon = -0.1$, we then tried increasing the belief for terms held in common. Again, the best performance was obtained when 10% of the available gain was added which gives the performance shown in Table 8.32.

Since the terms in nearest neighbor clusters appeared to be better descriptors than those from citation clusters, the performance of nearest neighbor links is surprising. Overall performance improves slightly for probabilistic and combined queries and degrades slightly for Boolean queries, although none of these changes

Table 8.32: Effect of nearest neighbor clusters on retrieval performance

	Precision (% change) – 50 queries					
	Probabilistic		Boolean		Combined	
Recall	baseline	with nn	baseline	with nn	baseline	with nn
10	67.2	67.5 (+0.3)	67.6	67.3 (−0.5)	76.2	76.8 (+0.9)
20	56.2	56.3 (+0.3)	58.8	58.2 (−1.0)	66.3	66.8 (+0.8)
30	47.6	47.4 (−0.3)	50.2	49.7 (−0.9)	56.3	56.2 (−0.1)
40	41.3	42.5 (+3.1)	45.2	44.8 (−0.8)	50.9	51.1 (+0.3)
50	34.4	35.5 (+3.0)	39.8	39.4 (−0.9)	45.4	46.1 (+1.7)
60	29.1	30.1 (+3.4)	33.6	33.4 (−0.6)	38.8	39.4 (+1.6)
70	20.3	20.2 (−0.6)	22.0	22.2 (+1.1)	25.1	25.4 (+1.1)
80	16.3	16.7 (+2.4)	18.1	18.2 (+0.7)	20.4	21.1 (+3.5)
90	11.3	11.4 (+0.4)	11.4	11.4 (−0.3)	12.4	12.5 (+0.8)
100	8.7	8.6 (−1.6)	7.9	7.8 (−1.1)	9.2	9.1 (−1.0)
average	33.3	33.6 (+1.1)	35.4	35.2 (−0.6)	40.1	40.4 (+0.9)

Table 8.33: Relevance of documents with citation and nearest neighbor links

	citations	nearest neighbors
Relevance judgements in collection	792	792
Relevance judgements with links	634 (80%)	297 (38%)
Relevant documents in collection	554	554
Relevant documents with links	431 (78%)	221 (40%)
Documents in collection with links	1747 (54%)	1928 (60%)
Number of links defined	6063	3317
Links per document in collection	1.9	1.0
Links per document with links	3.5	1.7

is significant. The main reason that the performance of nearest neighbor clusters is worse than citations appears to be because most relevant documents (and relevance judgements) do not have nearest neighbors but most relevant documents do have citation links (see Table 8.33). Although more documents have nearest neighbor links than have citations, a relevant document is twice as likely to have a citation as a nearest neighbor link. As a result, the apparently weak citation evidence has more

influence on the document ranking than the apparently stronger nearest neighbor evidence.

Documents that have nearest neighbors are much less likely to be judged relevant than a randomly selected document. Since the presence of a nearest neighbor link tends to raise belief in the original document, the effect of nearest neighbor links is to raise belief in non-relevant documents, thereby depressing performance. Given the earlier observation that short documents seem to be involved in nearest neighbor clusters, it is possible that the nearest neighbor algorithm used favors short documents and that these documents tend not to be judged relevant.

While nearest neighbors do not work well in this context, it is still possible that they could be useful for relevance feedback. If the evidence links were added only after a document was judged relevant by a user, then the fact that nearest neighbors favor non-relevant documents would not be a factor. It is also possible that nearest neighbor information would improve performance if added after the initial ranking, but only to highly ranked documents.

Our results are similar to those of Croft, *et al* [CT89] who used citations and nearest neighbor links in a spreading activation network. They also found that citations represented stronger evidence than nearest neighbor links for the CACM collection. They tested nearest neighbor links on five additional collections and found that performance improvements were possible, but that the improvements were small and inconsistent across collections. Finally, we found that using the nearest neighbor weight to estimate how much to increase belief in common terms did not improve performance which confirms their results.

Based on the citation results, we can accept Hypothesis 4. The negative results for nearest neighbor clusters do not suggest that these kinds of evidential links do

not improve performance. The nearest neighbor results show that nearest neighbor clustering, at least as implemented here, is not a good source of evidence.

8.7 Summary of results

Table 8.34: Network model performance improvement

	Precision (% change) – 50 queries			
Recall	probabilistic	NL	BL1	combined
10	60.2	69.6 (+15.7)	68.1 (+13.1)	77.4 (+28.6)
20	48.3	59.5 (+23.3)	60.1 (+24.6)	69.5 (+44.0)
30	41.0	49.8 (+21.5)	51.5 (+25.6)	58.9 (+43.6)
40	30.9	44.7 (+44.6)	46.8 (+51.6)	52.2 (+68.9)
50	26.5	39.0 (+47.5)	40.8 (+54.1)	47.6 (+79.8)
60	21.6	31.8 (+47.2)	34.5 (+59.8)	40.7 (+88.5)
70	15.0	21.9 (+46.2)	24.4 (+62.5)	28.0 (+86.8)
80	11.7	18.5 (+57.8)	19.6 (+66.9)	22.3 (+89.8)
90	6.4	12.4 (+93.2)	12.1 (+87.6)	13.7(+113.3)
100	4.4	9.4(+114.1)	8.4 (+92.8)	9.7(+121.7)
average	26.6	35.7 (+34.1)	36.6 (+37.7)	42.0 (+57.9)

Table 8.35: Average precision for p-norm and inference network Booleans

	Average Precision – 3 recall points			
	Best p-norm	Network Boolean	Boolean+ Citation	NL+Boolean +Citation
CACM	33.1 (p=2)	38.1 (+15.1%)–BL1 35.6 (+7.8%)–BL2	39.5 (+19.3%)	45.7 (+38.1%)
CISI	18.4 (p=1)	19.2 (+4.3%)		

Using the best features of the network model (basic model plus citations) gives the performance improvements shown in Table 8.34 when compared to a conventional probabilistic search. For natural language queries, average precision improves

34.1%, for BL1 it improves 37.7%, and for combined queries it improves by 57.9%. Comparing our results to reported p-norm results (Table 8.35) shows a 38.1% improvement over the best p-norm results. That these performance levels can be achieved at a computational cost that is comparable to conventional probabilistic retrieval clearly demonstrates the importance of the inference network model.

To summarize the results for the Hypotheses discussed in this chapter:

1. Given equivalent document representations, query forms, and assumptions about the match between indexing and query vocabularies, the inference network model will perform as well as probabilistic models.

   Hypothesis accepted – the inference network model significantly improves performance.

2. The use of multiple document representations will significantly improve retrieval performance when compared to equivalent networks without the additional representation.

   Our experimental results do not support any conclusion with regard to multiple document representations. The addition of CR categories can improve performance, but the improvement is inconsistent. The inconclusive results are due, at least in part, to problems with the original assignment of CR categories in the CACM collection.

3. The use of multiple query representations will significantly improve retrieval performance when compared to equivalent networks with a single query representation.

   Hypothesis accepted – significant improvements are consistently achieved.

4. The incorporation of dependencies between documents (citations and nearest neighbor links) will significantly improve retrieval performance when compared to equivalent networks without the additional dependencies.

   Hypothesis accepted on the basis of improvements resulting from the addition of citations.

The results described here also support a number of additional conclusions:

- Conventional probabilistic retrieval strategies based on *tf* and *idf* work well in the network model.
- The use of a non-zero default probability for term belief significantly improves performance.
- The use of query term weights based on the frequency of the term in the query improves performance for natural language queries.
- Even simple Boolean queries perform as well as corresponding probabilistic versions given the combining functions of equations 4.1 through 4.9.
- Effective Boolean query representations can be generated from a natural language query based on user supplied information about important terms and phrases.

Perhaps the most important conclusion of the current research is that evidence from multiple sources can be combined to improve our estimate of the probability that a document satisfies a query. As a result, the network model provides a natural framework within which to integrate results from several areas of information retrieval research (e.g., intelligent interfaces or NLP techniques).

Chapter 9

Implementation

Since the evaluation of Bayesian inference networks is known to be *NP*-complete [Coo90b], it was clear from the outset that simplifying assumptions would have to be made in order to use these techniques for anything beyond toy examples. In order for the results of this research to be useful, it must be possible to apply the inference techniques to collections of at least a million documents, which implies networks containing tens of millions of nodes. As a result, computational efficiency has been a major objective which led to our final hypothesis:

> Hypothesis 5 – It will be possible to build and evaluate these networks in "reasonable" time. In this context we will interpret "reasonable" to mean that for a collection with t term occurrences a) $O(t^2)$ time to build and preevaluate the network, b) query evaluation time relatively independent of document network size and less than 10 seconds for 90% of the queries when run on a typical micro-Vax or Sun network, and c) storage overhead that is a small constant ($c < 5$) times the original collection size.

In this chapter we will review the simplifications made to achieve our performance objectives (Section 9.1), describe how these networks are built and evaluated (Sections 9.2 through 9.4), and review the performance of the current implementation (Section 9.5).

9.1 Network simplifications

To ensure that retrieval networks are computationally tractable, we restrict them in two ways. First, we consider only the canonical link matrix forms of Section 4.5 which occupy $O(n)$ space and can be evaluated in $O(n)$ time, where n is the number of parents for a given node. These link matrix forms can be used to simulate the weighting schemes used in conventional retrieval models and experimental results do not suggest that more complex functions are required.

Second, we attach evidence only to the roots of the network. In terms of a network implementation, this restriction assures that we can propagate the effect of new evidence in a single pass through the network, visiting each node at most once. However, this restriction coupled with the fact that the structure of the document network is not modified during query processing means that $P(t_i|d_j)$ is completely determined by the belief values at the document nodes and we can use conventional inverted file techniques to implement the model. Without this restriction, $P(t_i|d_j)$ would be affected by evidence attached to nodes lower in the network and simple inverted file techniques could not be used.

9.2 Building the network

Returning to the network of Figure 4.1, we see that the only part of the document network that is visible to the query network is the set of representation concept nodes. For the purposes of query evaluation, all we need to know about the document network is the values of $P(t_i|d_j)$ for all document/term pairs. The basic idea then, is to build a document network, instantiate each document in

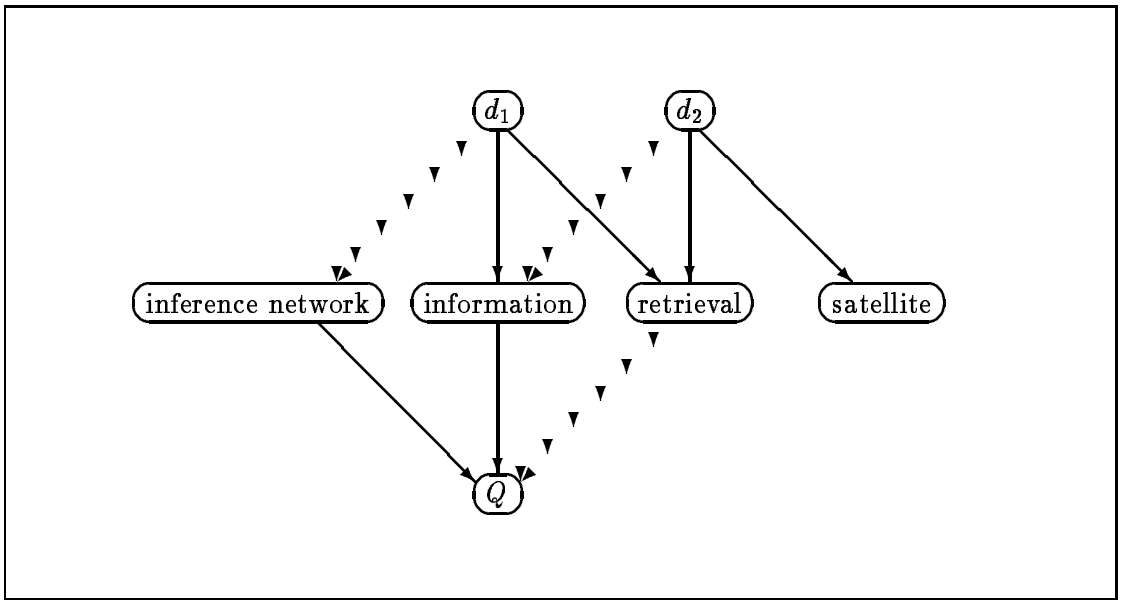

Figure 9.1: Inference network fragment

inference network	(d_1,0.731)	
information	(d_1,0.554)	(d_2,0.545)
retrieval	(d_1,0.554)	(d_2,0.715)
satellite	(d_1,0.665)	

Figure 9.2: Inverted belief lists

turn and record the belief in each representation concept given each document. This information can then be used to build a conventional inverted file in which we record, for each representation concept, a list of its belief values given each document in the collection. Fortunately, most documents change belief in only a small number of representation concepts, so for each representation concept we need only record this default belief (no parents instantiated) and a list of document/belief pairs for all documents that do affect belief when instantiated.

Repeating our example of Chapter 5, we found that in the network of Figure 9.1, instantiating d_1 resulted in

bel(inference network)	=	0.731	bel(information)	=	0.554
bel(retrieval)	=	0.554	bel(satellite)	=	0.000

and instantiating d_2 resulted in

bel(inference network)	=	0.000	bel(information)	=	0.545
bel(retrieval)	=	0.715	bel(satellite)	=	0.665

Using this information, we can build an inverted belief file containing the four records shown in Figure 9.2 and this file has all the information we need to evaluate queries (if an *idf* weighted default was used, then this would also be stored for each term).

For early experiments we built a general-purpose network evaluation package. Given a network described using a standard language, this package built and initialized the network and then computed changes to the network as evidence was attached. While this package was useful for initial experimentation, actually building a network and instantiating (and de-instantiating) each document in turn was far too slow to be useful – building the CACM collection with this package required in excess of 24 hours on an Intel 286 machine. Even with the assumption that all evidence is attached at the top, this approach evaluated each representation concept node for each document in which it occurred, and this evaluation generally required that the node be read from secondary storage.

A specialized package was then created to generate the inverted belief file from indexing transactions (one transaction per document/concept pair). This package is quite similar to those used to build commercial retrieval collections. By sorting the transactions to group all transactions for a given concept together, it is possible to efficiently build the inverted belief file without consulting the network description. With one additional optimization (using a hash table rather than a keyed file to look up concept identifiers), the time to build the CACM collection was reduced

to under two hours – most of the savings are due to improved times for generating inverted belief files.

This simple approach works for the basic model, but must be extended to handle document-document dependencies (citations and nearest neighbors) and additional term dependencies. For document-document dependencies a separate set of transactions is generated containing a transaction for each representation concept that is to be added to or altered in the set of concepts describing a document (e.g., for citations, a transaction for each term contained in a cited document). When building the inverted belief file, these two transaction sets are used to create the belief records for each concept. When multiple dependency types are to be added (e.g., citation and nearest neighbor), separate transaction sets are created and merged prior to building the inverted file. Similar extensions are required for addition of term dependencies, but were not required for the current research. All of these techniques can be readily extended to very large collections.

Building a document network for the basic model, then, involves four steps:

1. Generate a set of indexing transactions from the source texts.

2. Sort these transactions by concept and document identifier.

3. Build a network description.

4. Build an inverted belief file.

Step 3 is useful in a research setting since is simplifies many analysis steps (it is also required for the extended model), but could be omitted in a production setting using only the basic model – the only essential function it provides is as a dictionary of names for documents and representation concept names.

Table 9.1: File building times

	Elapsed time	% total	% 3-step total
generate transactions	4:31	30.7	69.1
sort	:52	5.9	13.3
build network description	8:11	55.6	
build inverted belief file	1:09	7.8	17.6
total	14:43	100.0	100.0

The growth of computational costs for steps 1, 3, and 4 is linear in the number of term occurrences (t) in the collection (strictly speaking, steps 3 and 4 are linear in the number of postings if term position information is not used, however, the number of postings is bounded by the number of term occurrences). Step 2 is $O(t \log t)$ for reasonable sorts, so the time to build a document network is $O(t \log t)$. The time required for these four steps when building the CACM collection on a Vax 6410 are shown in Table 9.1. The programs to generate transactions from the source text and to build the network description have not been optimized and could be improved substantially. If the network description is not required, then the CACM collection can be built in 6:32 or 0.12 seconds per record. Less than 15% of the time is in the sort, which will dominate for large collections. If the network description is required, then the collection can be build in 14:43 or 0.28 seconds per record with the sort accounting for roughly 6% of the total time.

The addition of new dependency types involves additional steps, but as long as the number of postings added is bounded by some constant times the number of original postings, the process remains $O(t \log t)$. While these additional dependencies will generally not change the order of computational costs, they can increase time and space requirements by significant (albeit constant) factors. It is also quite possible to imagine dependency structures that would not satisfy the constant bound (indeed, theoretically the worst case number of added citation postings is $O(t^2)$ –

all documents consist of a single term and cite all other documents – in practice, the number of citations per document is small and bounded).

The use of inverted file techniques to represent an inference network is a strategy that can be used when it is possible to predict the set of evidence assignments of interest, to predict the set of nodes whose belief we wish to assess, and the size of these sets has some reasonable bound.

If, for a binary-valued network, it is not possible to restrict these sets then there are 2^n evidence assignments of potential interest and the inverted file might contain as many as $n \times 2^n$ entries. However, for a retrieval network containing d documents and t representation concepts, there are only d evidence assignments of interest and for each assignment there are at most t belief values of interest, so at most

$$t \times d \leq \frac{n^2}{4}$$

belief values must be recorded. In practice, the number is much smaller. For CACM, where $d = 3204$ and $t = 5493$, we actually store only 79243 or $t \times 14.4$ belief values. For CISI, where $d = 1460$ and $t = 5448$ we store $t \times 13.0 = 71017$ belief values. The important factor here is clearly the number of evidence assignments of interest.

9.3 Space requirements

Table 9.2 shows the relevant file sizes for the test collections. Note that the source files include information besides the source text (citations, time and date entered).

The transaction files are generated in order to build the inverted belief files and need not be retained after the belief file is built. The three files that make up the document network are then the network description file, the dictionary (which converts terms to internal identifiers), and the inverted belief file. For CACM

	CACM	CISI
source file	2.3	2.2
indexing transactions	1.5	1.4
network description	2.1	2.0
dictionary	0.1	0.1
inverted belief file	0.6	0.6
citation transactions	2.2	
nearest neighbor transactions	0.5	
inverted belief(with citations)	1.5	

Table 9.2: File sizes (in megabytes)

these files total 2.8 megabytes which is 1.2 times the size of the source file. For CISI these files total 2.7 megabytes, which is also 1.2 times the size of the source file. The network description files used here are larger than necessary since they include belief information which is replicated in the inverted belief file. The current implementation does not display articles, so a production version would require an additional file containing the source text. If the unneeded information was eliminated from the inverted belief file and the source text was added, the set of files would be close to two times the size of the original source. The growth of these files will be linear or slightly sublinear in the size of the source text.

The use of citations noticeably increases the size of the inverted belief file. With citations included, the total size of the relevant files is 3.7 megabytes which is 1.6 times the size of the source file.

9.4 Evaluating queries

As with the document network, for our initial experiments with query evaluation we built complete query networks, attached them to document networks and instantiated the network. Given the inverted belief files and the fact that we are interested

in a very narrow range of query network topologies (probabilistic, Boolean, and combined queries) we can again improve efficiency.

It is difficult to develop useful computational bounds for query evaluation since performance depends heavily on the specific query and the terms that it contains. We will discuss the major efficiency considerations here, but will not develop theoretical bounds. Actual performance with the test set queries is a more useful performance indication.

Perhaps the single most important factor in query evaluation performance is the number of documents that must be examined to evaluate a term. Techniques that must examine every document in the collection to evaluate a term are usable only with test collections. In order to be practical for large collections, term evaluation should involve only those documents that contain the term.

A closely related factor is how the number of documents that must be examined grows with the size of the query. In this regard, probabilistic models and the inference network model (and most models that produce a ranking) are at a disadvantage relative to the conventional Boolean model. In general, the number of documents that must be considered by the ranking models increases monotonically with the number of terms evaluated, and the final result will contain the union of all documents contained in the inverted lists for each query term. Queries can be optimized in many ways (e.g., take small lists first, keep scores in a hash table [Dos82], or stop when new terms can no longer change the ranking [BL85]) but as the number of terms increases, so does the number of documents. Conventional Boolean queries can be optimized to order *and* and *and not* evaluations to minimize the number of documents which must be considered. Worst-case performance for the ranking and Boolean models is the same, an n-term *or* considers the same

number of documents as the ranking models; optimization can only be used to improve average performance.

For small collections or small queries the difference between ranking model and Boolean performance is not significant, but for very large collections or long queries the performance difference can be pronounced. At the same time, the importance of ranking increases for large collections. For large collections, considerable skill (and a long query) is required to retrieve a set of manageable size with a conventional Boolean query. If a ranking model works well, the user is generally not concerned with the size of the set as long as relevant documents appear at the top of the ranking.

In terms of query processing, then, the network and probabilistic models fall in the same general class. They perform substantially better with large collections than techniques which must examine the entire document space and somewhat worse than conventional Boolean techniques.

Given the closed-form expressions of Section 4.5, query evaluation is straightforward. The only unusual aspect is that a default estimate is computed as the query is evaluated. This default value is the value that would be assigned to a document that has contained none of the previously evaluated terms and is the correct value to be assigned as the initial value to documents that are introduced by the next term.

When all of the terms have been processed and the results have been normalized, the normalized default is the probability that a document which contains none of the query terms satisfies the query – all documents in the collection that are not found in the merged evaluation list are tied and have this default belief. Note that the default belief need not be lower than any in the evaluation list. Documents

can score lower than the default when a Boolean *not* is used (or when a degenerate estimate has been used somewhere in the network).

When the query is evaluated, in the current implementation we sort the entire evaluation list so that we can assign ranks to all documents in the collection. For large evaluation lists, this sort would be time consuming. Unless a full ranking is needed, there are faster ways (at worst $O(n)$) to pick some fixed number of documents from a large list than sorting it.

Using the inverted belief lists of Figure 9.2 we can evaluate a natural language query containing the phrase *inference network* and the terms *information* and *retrieval*. We will assume a fixed default belief of 0.4. We first merge the lists for *inference network* and *information*, adding beliefs for documents in both lists (equation 4.15) and adding the default for documents in a single list. This gives a combined list of

$$(d_1,1.285)\ (d_2,0.945)$$

and a combined default of 0.8. We then add the list for *retrieval* and normalize by dividing by the number of terms (we are assuming unweighted queries) to get

$$(d_1,0.613)\ (d_2,0.553)$$

with a default of 0.4.

To evaluate a Boolean conjunction, we first merge the lists for *inference network* and *information*, this time using the product form for combining beliefs (equation 4.1). This gives a combined list of

$$(d_1,0.405)\ (d_2,0.218)$$

and a combined default of 0.16. We then merge with the list for *retrieval* to get

$$(d_1,0.224)\ (d_2,0.156)$$

with a default of 0.064.

For the test collection queries running on a Vax 6410, the prototype processes the 50 CACM queries in 28 seconds for an average of 0.6 seconds per query for the natural language versions, 13 seconds or 0.3 seconds per query for BL1, 12 seconds or 0.2 seconds per query for BL2, and 34 seconds or 0.7 seconds per query for the combined queries. The 35 CISI queries take 11 seconds or 0.3 seconds per query for the natural language versions, 9 seconds or 0.3 seconds per query for the Boolean, and 17 seconds or 0.5 seconds per query for the combined queries. This performance is well within the hypothesized bound of ten seconds. The current network implementation makes no attempt to optimize query evaluation and captures information that is not needed for an operational system, so these times could be improved.

Query evaluation time is not, however, entirely independent of network size since it depends upon the size of the posting lists in the inverted file which grow with the size of the collection. The rate of growth for the average posting list is logarithmic in the size of the collection, so average query evaluation time is "nearly" independent.

9.5 Summary

Given the performance characteristics just described, we can accept Hypothesis 5. Query networks can be build in $O(t \log t)$ time (which is certainly $O((n+t)^2)$), queries can be processed within the hypothesized time bound for reasonable collection sizes (although evaluation time is not completely independent of network size), and the files occupy substantially less than five times the size of the source text and exhibit linear or slightly sublinear growth.

While it is certainly possible to imagine network features that would degrade performance, with the current model it is possible to significantly improve retrieval performance at computational costs that are quite similar to conventional probabilistic systems and that are practical for large collections.

Chapter 10

Conclusions and Future Research

In this dissertation we have described a new formal retrieval model, showed that it generalizes existing retrieval models, and have presented experimental results which demonstrate significant improvements in retrieval performance when compared to conventional models. In this final chapter we review the major contributions of this research (Section 10.1) and describe the major areas where additional research should be pursued (Section 10.2).

10.1 Contributions

This research represents a significant contribution to information retrieval theory and has developed retrieval techniques that can considerably improve performance of commercial retrieval systems. The specific contributions of this work are:

- Very significant improvements in retrieval performance over conventional retrieval models.
- Development of a formal retrieval model that generalizes probabilistic, Boolean, extended Boolean, and cluster-based retrieval models.

- Development of a theoretical framework that allows the results from several different retrieval techniques to be combined when assessing the probability that a document matches a query.

- Development of a mechanism for representing uncertain information needs. Representation of uncertainty is a very general problem that must be dealt with in many automated tasks.

- Development of techniques for representing complex information objects. Documents (and text in general) represent a particularly important class of complex object, and the techniques for integrating multiple representations of objects are of general interest.

- Development of techniques for efficient evaluation of a restricted class of probabilistic inference networks.

- This research represents a step toward more integrated information systems. The retrieval model can be integrated with a number of other types of systems (e.g., DBMS, hypertext, office automation) to improve text handling.

10.2 Future research

We have accomplished most of the objectives originally established for this research and have demonstrated the theoretical and practical utility of the inference network retrieval model. However, one of our original Hypotheses remains open and the work reported here has suggested several additional areas where work is required.

The one Hypothesis that we were unable to test dealt with the use of multiple document representations. Given the strong body of evidence that suggests that

multiple representations represent relatively independent sources of information about document content and can be used to retrieve different documents for the same query, we believe that the inconsistent performance observed results from problems with the assignment of Computing Reviews categories to the CACM collection. Additional experiments will be required using a test collection with a richer set of document representations. Two collections are potentially interesting for these tests. One is a subset of the National Library of Medicine collection which includes manually assigned Medical Subject Heading (MeSH) descriptors. The second is a legal database which contains manually assigned subject codes and links to related opinions and statutes.

Additional areas where further work is required are:

- Work with additional NLP-based document and query representations currently under development. Other researchers at the University of Massachusetts are developing new representation techniques based on phrases and disambiguated terms. Work will be required to determine how best to integrate these new forms of evidence.

- Replicate experiments on other test collections. The network model has shown consistent improvements on two standard collections. While this represents strong evidence that the model improves retrieval performance, these test should be repeated on additional test collections.

- Incorporate term dependencies in the network model. The current model makes only limited use of term dependence information (phrases and thesaurus information). We need to implement and test the effect of thesaurus information and should extend the model to incorporate additional dependencies (e.g., term clustering).

- Implement and test relevance feedback. The relevance feedback model of Section 7.1 has not been tested. Since relevance feedback gives very significant improvements with conventional probabilistic models and is generally more effective when the initial query produces a good ranking, we expect significant improvements in network model performance.

- Test the model with large full-text collections. While evaluation with test collections is important, if these techniques are to have an commercial impact, they must be evaluated on very large collections.

- Test the effect of user supplied information about query and term importance. Our work with user supplied information about phrase structure and term importance suggests that this information can be used to improve retrieval performance. Our experiments are, however, preliminary and additional work will be required to determine how best to obtain this information and how much improvement is possible.

- Experiment with ways to learn better belief estimates. The current document network makes no attempt to improve the dependency estimates over time. It is possible that better estimates could be learned from a large sample of queries and relevance judgements (as in document space modification).

A final area of further research would explore the nature of the interactions between the different types of evidence available for document retrieval. As a result of this work we know that some forms of evidence complement each other and improve performance, while combining other forms of evidence gives no performance improvement or actually degrades performance. Presumably, the nature of the interaction is related to the degree that the evidence forms are "independent," but

further work will be required to better define this notion of independence and to determine how it can be measured.

APPENDIX

TENSOR NOTATION FOR UPDATE RULES

While we follow Pearl in referring to the conditional probabilities stored at each node as link matrices, they are really tensors. The matrix notation is workable for nodes with up to two parents since vectors and matrices are first and second order tensors, but the matrix notation is cumbersome for nodes with more than two parents. To see why tensor notation is required, consider a node d with a single parent a. Both nodes take on values from the set $\{v_1, \ldots, v_m\}$. To specify $P(d|a)$ we need to provide values for

$$P(d = v_1|a = v_1), P(d = v_1|a = v_2), \ldots, p(d = v_m|a = v_m)$$

which can be represented by an $m \times m$ matrix. If we now add a parent b we must specify $m \times m \times m$ values, but this is no longer a matrix. Essentially, in order to add the second parent we create an m element vector whose components are $m \times m$ matrices but this is, in fact, a third-order tensor. Similarly, to add a third parent we need a vector whose elements are third-order tensors, and so on. For a node with n parents we must specify m^{n+1} values and would like to do so using a notation that makes the indices for each node explicit. Thus, for a three parent example, we have a link tensor L_{abc} where each of the subscripts represents indices for a single parent.

In this appendix we reformulate Pearl's propagation rules for causal polytrees (see [Pea88, pages 182-184]) using tensor notation. The use of tensor notation

clarifies the computational aspects of networks containing nodes with more than two parents.

The probability specification for a node X, which can assume d discrete values, conditioned on its parents $U_1, \ldots, U_n$ is a d-dimension tensor L of order $n+1$, that is,

$$P(x|u_1, \ldots, u_n) = L_{x,u_1,\ldots,u_n}.$$

We will call L the link tensor. The link tensor is used to compute the nodal π vector which summarizes the predictive evidence available at a node and to compute the likelihood vectors to be sent by X to its parents.

The vector π_{u_i} provided by parent U_i to X specifies the probability that $X = x$ conditioned on the evidence available at U_i (π_{u_i} is Pearl's $\pi_X(u_i)$). Pearl's method for computing the nodal π vector [Pea88, equation 4.51]

$$\pi(x) = \sum_{u_1,\ldots,u_n} P(x|u_1, \ldots, u_n) \prod_{1 \le i \le n} \pi_X(u_i)$$

is really the inner product of the link tensor and the outer product of the individual π_{u_i} vectors. If we denote this outer product by Q

$$\begin{aligned} \pi(x) &= L_{x,u_1,\ldots,u_n} \pi_{u_1} \pi_{u_2} \ldots \pi_{u_n} \\ &= L_{x,u_1,\ldots,u_n} Q_{u_1,u_2,\ldots,u_n} \\ &= R_{x,u_1,\ldots,u_n,u_1,u_2,\ldots,u_n}. \end{aligned}$$

R is contracted on the common parent indices to give the vector

$$\pi(x) = R_x.$$

To avoid the d^{2n+1} element intermediate result we can form the product and contract on common indices one parent at a time

$$\pi(x) = L_{x,u_1,\ldots,u_n} \pi_{u_1} \ldots \pi_{u_n}$$

$$
\begin{aligned}
&= R^{n-1}_{x,u_1,\ldots,u_{n-1}} \pi_{u_1} \ldots \pi_{u_{n-1}} \\
&= R^{n-2}_{x,u_1,\ldots,u_{n-2}} \pi_{u_1} \ldots \pi_{u_{n-2}} \\
&\vdots \\
&= R^{1}_{x,u_1} \pi_{u_1} \\
&= R_x.
\end{aligned}
$$

Note that superscripts are used to identify unique tensors, not to distinguish covariant and contravariant indices.

The procedure for computing likelihood vectors for parents [Pea88, equation 4.52] is similar except the outer product used to compute the λ_{u_i} vector for U_i excludes the π_{u_i} vector received from U_i. Thus, we are forming an inner product of the order $n+1$ link tensor and the order $n-1$ outer product of the π_{u_i} vectors to get an order 2 result. This result is then multiplied by the nodal likelihood vector λ and contracted on x to obtain λ_{u_i}. Again, the product/contraction can be performed one parent at a time. This gives

$$
\begin{aligned}
\lambda_{u_i} &= \lambda L_{x,u_1,\ldots,u_n} \pi_{u_1} \ldots \pi_{u_{i-1}} \pi_{u_{i+1}} \ldots \pi_{u_n} \\
&= \lambda R^{n-1}_{x,u_1,\ldots,u_{n-1}} \pi_{u_1} \ldots \pi_{u_{i-1}} \pi_{u_{i+1}} \ldots \pi_{u_{n-1}} \\
&\vdots \\
&= \lambda R^{1}_{x_1,u_1,u_i} \pi_{u_1} \\
&= \lambda R_{x,u_i} \\
&= S_{u_i}
\end{aligned}
$$

which is the likelihood vector for U_i. Forming the product and contracting one parent at a time is particularly easy to implement (and understand) and operates in space bounded by the size of L.

References

[All87] Allen, James. *Natural Language Understanding.* Benjamin/Cummings, Menlo Park, CA, 1987.

[Ang75] Angione, Pauline. On the equivalence of Boolean and weighted searching based on the convertibility of query forms. *Journal of the American Society for Information Science*, 27(2):112–124, March 1975.

[AOJJ89] Andersen, Stig K., Olesen, Kristian G., Jensen, Finn V., and Jensen, Frank. HUGIN–a shell for building Bayesian belief universes for expert systems. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1080–1085, August 1989.

[Baa88] Baase, Sara. *Computer Algorithms: Introduction to Design and Analysis.* Addison Wesley, 1988.

[BC87] Belkin, Nicholas J. and Croft, W. Bruce. Retrieval techniques. In Williams, Martha E., editor, *Annual Review of Information Science and Technology*, chapter 4, pages 109–145. Elsevier Science Publishers, 1987.

[Bel89] Belew, Richard K. Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. In Belkin, N. J. and van Rijsbergen, C. J., editors, *Proceedings of the* 12^{th} *Annual International Conference on Research and Development in Information Retrieval*, pages 11–20, New York, NY, 1989. ACM.

[BK86] Belkin, Nicholas J. and Kwasnik, B. H. Using structural representations of anomalous states of knowledge for choosing document retrieval strategies. In Rabitti, Fausto, editor, *Proceedings of the* 9^{th} *Annual Internationsl Conference on Research and Development in Information Retrieval*, pages 11–22, 1986.

[BL85] Buckley, Chris and Lewit, Alan F. Optimization of inverted vector searches. In *Proceedings of the Eighth Annual International Conference on Research and Development in Information Retrieval*, pages 97–110, New York, NY, 1985. ACM.

[Bla88] Blair, David C. An extended relational retrieval model. *Information Processing and Management*, 24(3):349–371, 1988.

[BM85] Blair, D. C. and Maron, M. E. An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of the ACM*, 28(3):290–299, 1985.

[BM88] Brachman, Ronald J. and McGuiness, Deborah L. Knowledge representation, connectionism, and conceptual retrieval. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 161–174, New York, NY, 1988. ACM.

[BOB82a] Belkin, N. J., Oddy, R. N., and Brooks, H. M. ASK for information retrieval: Part I. Background and theory. *Journal of Documentation*, 38(2):61–71, June 1982.

[BOB82b] Belkin, N. J., Oddy, R. N., and Brooks, H. M. ASK for information retrieval: Part II. Results of a design study. *Journal of Documentation*, 38(3):145–164, September 1982.

[Boo85] Bookstein, Abraham. Probability and fuzzy-set applications to information retrieval. In Williams, Martha E., editor, *Annual Review of Information Science and Technology*, pages 117–151. Knowledge Industries Publications, Inc., 1985.

[Boo89] Bookstein, Abraham. Set-oriented retrieval. *Information Processing and Management*, 25(5):465–475, 1989.

[BS75] Bookstein, Abraham and Swanson, Donald R. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science*, 26(1):45–50, 1975.

[CCN90] Crouch, Carolyn J., Crouch, Donald B., and Nareddy, Krishna R. The automatic generation of extended queries. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 369–383. ACM, September 1990.

[CD90] Croft, W. Bruce and Das, Raj. Experiments with query acquisition and use in document retrieval systems. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 349–368. ACM, September 1990.

[CF89] Chang, Kuo-Chu and Fung, Robert. Node aggregation for distributed inference in Bayesian networks. In *IJCAI 1989*, pages 265–270, 1989.

[CH79] Croft, W. Bruce and Harper, D. J. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35:285–295, 1979.

[Che88] Cheeseman, Peter. An inquiry into computer understanding. *Computational Intelligence*, 4:58–66, February 1988.

[CK87] Cohen, Paul R. and Kjeldsen, Rick. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23(2):255–268, 1987.

[CL68] Chow, C. K. and Liu, C. N. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[CLCW89] Croft, W. Bruce, Lucia, T. J., Cringean, J., and Willett, P. Retrieving documents by plausible inference: An experimental study. *Information Processing and Management*, 25(6):599–614, 1989.

[CM78] Cooper, W. S. and Maron, M. E. Foundations of probabilistic and utility-theoretic indexing. *Journal of the ACM*, 25(1):67–80, January 1978.

[Coh85] Cohen, Paul R. *Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach.* Pitman, Boston, MA, 1985.

[Coo71] Cooper, W. S. A definition of relevance for information retrieval. *Information Storage and Retrieval*, 7:19–37, 1971.

[Coo90a] Coombs, James H. Hypertext, full text, and automatic linking. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 83–98. ACM, September 1990.

[Coo90b] Cooper, Gregory F. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, March 1990.

[CP85] Croft, W. Bruce and Parenty, Thomas J. A comparison of a network structure and a database system used for document retrieval. *Information Systems*, 10(4):377–390, 1985.

[Cro80] Croft, W. Bruce. A model of cluster searching based on classification. *Information Systems*, 5:189–195, 1980.

[Cro81] Croft, W. Bruce. Incorporating different search models into one document retrieval system. *SIGIR Forum*, 16(1):40–45, 1981.

[Cro84] Croft, W. Bruce. A comparison of the cosine correlation and the modified probabilistic model. *Information Technology*, 3(2):113–114, 1984.

[Cro86] Croft, W. Bruce. Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, 37(2):71–77, 1986.

[Cro87] Croft, W. Bruce. Approaches to intelligent information retrieval. *Information Processing and Management*, 23(4):249–254, 1987.

[CT84] Croft, W. Bruce and Thompson, Roger H. The use of adaptive mechanisms for selection of search strategies in document retrieval systems. In van Rijsbergen, C. J., editor, *Proceedings of the ACM/BCS International Conference on Research and Development in Information Retrieval*, pages 95–110, 1984.

[CT87] Croft, W. Bruce and Thompson, Roger H. I^3R: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6):389–404, November 1987.

[CT89] Croft, W. Bruce and Turtle, Howard. A retrieval model incorporating hypertext links. In *Hypertext '89 Proceedings*, pages 213–224, 1989.

[CTS78] Cannings, C., Thompson, E. A., and Skolnick, M. H. Probability functions on complex pedigrees. *Advanced Applied Probability*, 10:26–61, 1978.

[DAR90] DARPA. Tipster information package. Information package related to Defense Advanced Research Project Agency BAA 90-16, July 1990.

[Dem68] Dempster, A. P. A generalization of Bayesian inference. *Journal of the Royal Statistical Society B*, 30:205–247, 1968.

[DHB+78] Duda, R. O., Hart, P. E., Barnett, P., Gaschnig, J., Konolige, K., Reboh, R., and Slocum, J. Development of the PROSPECTOR consultant system for mineral exploration. Technical report, SRI International Artificial Intelligence Center, 1978. Final report on SRI projects 5821 and 6915.

[DHN76] Duda, R. O., Hart, P. E., and Nilsson, N. J. Subjective Bayesian methods for rule-based inference. In *Proceedings of the National Computer Conference*, volume 45, pages 1075–1082, 1976.

[Dos82] Doszkocs, Tamas D. From research to application: the CITE natural language information retrieval system. In Salton, G. and Schneider, H. J., editors, *Research and Development in Information Retrieval*, pages 251–262. Springer-Verlag, 1982.

[Doy79] Doyle, John. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

[EW61] Edmundson, H. P. and Wyllys, R. E. Automatic abstracting and indexing survey and recommendations. *Communications of the ACM*, 4:226–234, 1961.

[Fag87] Fagan, Joel L. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-syntactic Methods.* PhD thesis, Cornell University, 1987.

[FB90] Fuhr, Norbert and Buckley, Chris. Probabilistic document indexing from relevance feedback data. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 45–61. ACM, September 1990.

[FC89] Frisse, Mark E. and Cousins, Steve B. Information retrieval from hypertext: Update on the dynamic medical handbook project. In *Hypertext '89 Proceedings*, pages 199–212, 1989.

[FCAT90] Fung, Robert M., Crawford, Stuart L., Applebaum, Lee A., and Tong, Richard M. An architecture for probabilistic concept-based information retrieval. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 455–467. ACM, September 1990.

[FLGD87] Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, November 1987.

[FNL88] Fox, Edward A., Nunn, Gary L., and Lee, Whay C. Coefficients for combining concept classes in a collection. In *Proceedings of the Eleventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–308, New York, NY, 1988. ACM.

[Fox83a] Fox, Edward A. Characterization of two new experimental collections in computer and information science containing textual and bibliographic concepts. Technical Report 83-561, Department of Computer Science, Cornell University, Ithaca, NY 14853, September 1983.

[Fox83b] Fox, Edward A. *Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types.* PhD thesis, Cornell University, 1983.

[Fox86] Fox, John. Three arguments for extending the framework of probability. In Kanal, Laveen N. and Lemmer, John F., editors, *Uncertainty in Artificial Intelligence*, pages 447–458. North-Holland, Amsterdam, 1986.

[Fuh89a] Fuhr, Norbert. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[Fuh89b] Fuhr, Norbert. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3):183–204, July 1989.

[GLF81] Garvey, Thomas D., Lowrance, John D., and Fischler, Martin A. An inference technique for integrating knowledge from disparate sources. In *Proceedings of the Seventh International Joint Conference in Artificial Intelligence*, pages 319–325, August 1981.

[GS84] Gordon, Jean and Shortliffe, Edward H. The Dempster-Shafer theory of evidence. In Buchannon and Shortliffe, editors, *Rule-based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*, chapter 13, pages 272–292. Addison-Wesley, 1984.

[GWDS86] Griffith, B. C., White, H. D., Drott, M. C., and Saye, J. D. Tests of methods of evaluating bibliographic databases: An analysis of the National Library of Medicine's handling of literatures in the medical behavioral sciences. *Journal of the American Society for Information Science*, 37(4):261–270, 1986.

[Hes55] Hessel, Alfred. *A History of Libraries.* The Scarecrow Press, New Brunswick, NJ, 1955. Translated by Reuben Peiss.

[Hur82] Hurt, C. D. A comparison of a bibliometric approach and an historical approach. *Information Processing and Management*, 19(3):151–157, 1982.

[JF87] Jones, W. P. and Furnas, G. W. Pictures of relevance – a geometric analysis of similarity measures. *Journal of the American Society for Information Science*, 38(6):420–442, 1987.

[KB84] Kraft, Donald H. and Buell, Duncan A. Fuzzy sets and generalized Boolean retrieval systems. *International Journal of Man-Machine Studies*, 19:45–56, 1984.

[KC89] Krovetz, Robert and Croft, W. Bruce. Word sense disambiguation using a machine readable dictionary. In Belkin, N. J. and van Rijsbergen, C. J., editors, *Proceedings of the 12th International Conference on Research and Development in Information Retrieval*, pages 127–136, June 1989.

[Kes65] Kessler, M. M. Comparison of the results of bibliographic coupling and analytic subject indexing. *American Documentation*, 16(3):223–233, 1965.

[KL86] Kanal, Laveen N. and Lemmer, John F., editors. *Uncertainty in Artificial Intelligence*. North-Holland, Amsterdam, 1986.

[KMT+82] Katzer, J., McGill, M. J., Tessier, J. A., Frakes, W., and DasGupta, P. A study of the overlap among document representations. *Information Technology: Research and Development*, 1:261–274, 1982.

[Kwo89] Kwok, K. L. A neural network for probabilistic information retrieval. In Belkin, N. J. and van Rijsbergen, C. J., editors, *Proceedings of the 12th International Conference on Research and Development in Information Retrieval*, pages 21–30, June 1989.

[LB88] Losee, Robert M. and Bookstein, Abraham. Integrating Boolean queries in Conjunctive Normal Form with probabilistic retrieval models. *Journal of the American Society for Information Science*, 24(3):315–321, 1988.

[LC90] Lewis, David D. and Croft, W. Bruce. Term clustering of syntactic phrases. In Vidick, Jean-Luc, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 385–404. ACM, September 1990.

[LCB89] Lewis, David, Croft, W. Bruce, and Bhandaru, Nehru. Language-oriented information retrieval. *International Journal of Intelligent Systems*, 1989. Forthcoming.

[Lew89] Lewis, David D. A description of CACM-3204-ML1, a test collection for information retrieval and machine learning. Technical Report IRL89-5, Information Retrieval Laboratory, Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, November 1989.

[Lew90] Lewis, David D. Representation, learning, and language in information retrieval. PhD dissertation proposal, University of Massachusetts, 1990.

[LGS86] Lowrance, John D., Garvey, Thomas D., and Strat, Thomas M. A framework for evidential reasoning systems. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 896–903, 1986.

[Lin87] Lindley, Dennis V. The probability approach to the treatment of uncertainty in artificial intelligence and expert systems. *Statistical Science*, 2(1):17–24, February 1987.

[LK88] Lemmer, John F. and Kanal, Laveen N., editors. *Uncertainty in Artificial Intelligence 2.* North-Holland, Amsterdam, 1988.

[LS88] Lauritzen, S. L. and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50(2):157–224, 1988.

[MC87] Monarch, I. and Carbonell, Jaime. CoalSORT: A knowledge-based interface. *IEEE Expert*, pages 39–53, 1987.

[MK60] Maron, M. E. and Kuhns, J. L. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7:216–244, 1960.

[MKN79] McGill, Michael, Koll, Mathew, and Noreault, Terry. An evaluation of factors affecting document ranking by information retrieval systems. Technical report, Syracuse University, School of Information Studies, 1979.

[Nil84] Nilsson, Nils J. Probabilistic logic. Technical Note 321, SRI International, February 1984.

[Nil86] Nilsson, Nils J. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[NKM77] Noreault, Terry, Koll, Mathew, and McGill, Michael J. Automatic ranked output for Boolean searches in SIRE. *Journal of the American Society for Information Science*, 28(6):333–339, 1977.

[Odd77] Oddy, Robert N. Information retrieval through man-machine dialogue. *Journal of Documentation*, 33:1–14, 1977.

[OPC86] Oddy, Robert N., Palmquist, Ruth A., and Crawford, Margaret A. Representation of anomalous states of knowledge in information retrieval. In *Proceedings of the 1986 ASIS Annual Conference*, pages 248–254, 1986.

[Pea88] Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, 1988.

[PF85] Pao, Miranda Lee and Fu, T. Titles retrieved from MEDLINE and from citation relations. *Proceedings of the American Society for Information Science*, 22:120–125, 1985.

[Pol87] Pollit, S. CANSEARCH: An expert systems approach to information retrieval. *Information Processing and Management*, 23(2):119–138, 1987.

[PP85] Pearl, Judea and Paz, Azaria. GRAPHOIDS: A graph-based logic for reasoning about relevance relations. Technical Report 850038(R-53-L), Cognitive Systems Laboratory, University of California, Los Angeles, 1985.

[PV87] Pearl, Judea and Verma, Thomas S. The logic of representing dependencies by directed graphs. In *Proceeding of the Sixth National Conference on Artificial Intelligence*, pages 374–379, 1987.

[PW89] Pao, Miranda Lee and Worthen, Dennis B. Retrieval effectiveness by semantic and citation searching. *Journal of the American Society for Information Science*, 40(2):226–235, 1989.

[Qui83] Quinlan, J. R. INFERNO: A cautious approach to uncertain inference. *The Computer Journal*, 26:255–269, 1983.

[Rad79] Radecki, Tadeusz. Fuzzy set theoretical approach to document retrieval. *Information Processing and Management*, 15:247–259, 1979.

[Rad83] Radecki, Tadeusz. Incorporation of relevance feedback into Boolean retrieval systems. In Salton, G. and Schneider, H. J., editors, *Research and Development in Information Retrieval*, pages 133–150, 1983.

[Rad88] Radecki, Tadeusz. Probabilistic methods for ranking output documents in conventional Boolean retrieval systems. *Information Processing and Management*, 24(3):281–302, 1988.

[RMC82] Robertson, S. E., Maron, M. E., and Cooper, W. S. Probability of relevance: A unification of two competing models for document retrieval. *Information Technology: Research and Development*, 1(1):1–21, January 1982.

[Rob77] Robertson, S. E. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, December 1977.

[Rob81] Roberston, Stephen E. The methodology of information retrieval experiment. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 1, pages 9–31. Butterworth, 1981.

[RS76] Robertson, S. E. and Sparck Jones, K. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, May-June 1976.

[Sal63] Salton, Gerard. Associative document retrieval techniques using bibliographic information. *Journal of the ACM*, 10(4):440–457, 1963.

[Sal68] Salton, Gerard. *Automatic Information Organization and Retrieval.* McGraw-Hill, 1968.

[Sal86] Salton, Gerard. On the use of knowledge-based processing in automatic text retrieval. In *Proceedings of the 1986 ASIS Annual Conference*, pages 277–287, 1986.

[Sal88] Salton, Gerard. A simple blueprint for automatic Boolean query processing. *Information Processing and Management*, 24(3):269–280, 1988.

[Sav54] Savage, Leonard J. *The Foundations of Statistics.* Wiley, New York, NY, 1954.

[SB77] Sparck Jones, Karen and Bates, R. G. Research on automatic indexing 1974–1976. Technical report, Computer Laboratory, University of Cambridge, 1977.

[SB87] Salton, Gerard and Buckley, Chris. Term weighting approaches in automatic text retrieval. Technical Report 87-881, Department of Computer Science, Cornell University, Ithaca, NY, November 1987.

[SB88] Salton, Gerard and Buckley, Chris. On the use of spreading activation methods in automatic information retrieval. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 147–160, New York, NY, 1988. ACM.

[Sch87] Schum, D. A. *Evidence and Inference for the Intelligence Analyst.* University Press of America, Lanham, MD, 1987.

[SFV83] Salton, G., Fox, E. A., and Voorhees, E. Advanced feedback methods in information retrieval. *Journal of the American Society for Information Science*, 36(3):200–210, 1983.

[SFW83] Salton, Gerard, Fox, Edward, and Wu, H. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[Sha76] Shafer, Glen. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Sha86] Shachter, Ross D. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, November-December 1986.

[Sha87a] Shafer, Glen. Belief functions and possibility measures. In Bezdek, J., editor, *Analysis of Fuzzy Information, Volume 1: Mathematics and Logic*, pages 51–58. CRC Press, Boca Raton, FL, 1987.

[Sha87b] Shafer, Glenn. Probability judgment in artificial intelligence and expert systems. *Statistical Science*, 2(1):3–16, February 1987.

[Sho76] Shortliffe, Edward H. *Computer-based Medical Consultation: MYCIN.* Elsevier, New York, NY, 1976.

[Sho85] Shoval, Peretz. Principles, procedures and rules in an expert system for information retrieval. *Information Processing and Management*, 21(6):475–487, 1985.

[Sie56] Siegel, Sydney. *Non-parametric Statistics for the Behavioral Sciences.* Mcgraw-Hill, 1956.

[SKJ84] Spiegelhalter, D. J. and Knill-Jones, R. P. Statistical and knowledge-based approaches to clinical decision-support systems, with an application in gastroenterology. *Journal of the Royal Statistical Society A*, 147:35–77, 1984.

[SM82] Schum, D. A. and Martin, A. W. Formal and empirical research on cascaded inference in jurisprudence. *Law and Society Review*, 17:105–157, 1982.

[SM83] Salton, Gerard and McGill, Michael J. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[Sma73] Small, H. Co-citation in scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24:265–269, 1973.

[SP90] Shafer, Glenn and Pearl, Judea, editors. *Uncertain Reasoning.* Morgan Kaufmann, San Mateo, CA, 1990.

[Spa71] Sparck Jones, Karen. *Automatic Keyword Classification for Information Retrieval.* Archon Books, 1971.

[Spa74] Sparck Jones, Karen. Automatic indexing. *Journal of Documentation*, 30(4):393–432, 1974.

[Spa81] Sparck Jones, Karen. Retrieval system tests 1958-1978. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 12, pages 213–255. Butterworth, 1981.

[Spi86] Spiegelhalter, David J. A statistical view of uncertainty in expert systems. In Gale, W., editor, *Artificial Intelligence and Statistics*, chapter 2, pages 17–55. Addison-Wesley, Reading, MA, 1986.

[Spi87] Spiegelhalter, David J. Probabilistic expert systems in medicine: Practical issues in handling uncertainty. *Statistical Science*, 2(1):25–29, February 1987.

[SSG89] Smith, Philip J., Shute, Steven J., and Galdes, Deb. Knowledge-based search tactics for an intelligent intermediary system. *ACM Transactions on Information Systems*, 7(3):246–270, July 1989.

[ST84] Sparck Jones, Karen and Tait, J. Automatic search term variant generation. *Journal of Documentation*, 40(1):50–66, 1984.

[Sti75] Stirling, K. H. The effect of document ranking on retrieval system performance: A search for an optimal ranking rule. *Proceedings of the American Society for Information Science*, 12:105–106, 1975.

[Sv76] Sparck Jones, Karen and van Rijsbergen, C. J. Information retrieval test collections. *Journal of Documentation*, 32(1):59–75, 1976.

[SVF84] Salton, G., Voorhees, E., and Fox, E. A. A comparison of two methods for Boolean query relevancy feedback. *Information Processing and Management*, 20(5/6):637–651, 1984.

[SYY75] Salton, G., Yang, C. S., and Yu, C. T. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975.

[TAAC86] Tong, Richard M., Applebaum, Lee A., Askman, Victor N., and Cunningham, James F. RUBRIC III: An object-oriented expert system for information retrieval. In Karna, Kamal N., Parsaye, Kamran, and Silverman, Barry G., editors, *Second Expert Systems in Government Symposium*, pages 106–115, 1986.

[Tag81] Tague, Jean M. The pragmatics of information retrieval experimentation. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 5, pages 59–102. Butterworth, 1981.

[TC89] Thompson, Roger H. and Croft, W. Bruce. Support for browsing in an intelligent text retrieval system. *International Journal of Man-Machine Studies*, 30:639–668, 1989.

[TS85] Tong, Richard M. and Shapiro, Daniel. Experimental investigations of uncertainty in a rule-based system for information retrieval. *International Journal of Man-Machine Studies*, 22:265–282, 1985.

[TSMD83] Tong, Richard M., Shapiro, Daniel G., McCune, Brian P., and Dean, Jeffrey S. A rule-based approach to information retrieval: Some results and comments. In *Proceedings of the National Conference on Artificial Intelligence*, pages 411–415, 1983.

[Tur90] Turtle, Howard. Attribute recall for known-item searching. Technical Report IR 90-2, Information Retrieval Laboratory, Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003, 1990.

[van79] van Rijsbergen, C. J. *Information Retrieval.* Butterworths, 1979.

[van86] van Rijsbergen, C. J. A non-classical logic for information retrieval. *Computer Journal*, 29(6):481–485, 1986.

[van89] van Rijsbergen, C. J. Towards an information logic. In Belkin, N. J. and van Rijsbergen, C. J., editors, *Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 77–86, New York, NY, 1989. ACM.

[Wil73] Wilson, Patrick. Situational relevance. *Information Storage and Retrieval*, 9:457–471, 1973.

[Wil88] Willett, Peter. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24(5):577–598, 1988.

[YM88] Yu, Clement T. and Mizuno, Hirotaka. Two learning schemes for information retrieval. In Chiaramella, Yves, editor, *Proceedings of the* 11^{th} *International Conference on Research and Development in Information Retrieval*, pages 201–218, 1988.

[Zad83] Zadeh, Lotfi A. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11:199–228, 1983.

[Zad86a] Zadeh, Lotfi A. Is probability theory sufficient for dealing with uncertainty in AI: A negative view. In Kanal, Laveen N. and Lemmer, John F., editors, *Uncertainty in Artificial Intelligence*, pages 103–116. North-Holland, Amsterdam, 1986.

[Zad86b] Zadeh, Lotfi A. A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination. *AI Magazine*, 7(2):81–90, 1986.