TREC and TIPSTER Experiments With INQUERY

James P. Callan, W. Bruce Croft and John Broglio
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610, USA

E-mail: {callan, croft, broglio}@cs.umass.edu

July 8, 1994

Abstract

INQUERY is a probablistic information retrieval system based upon a Bayesian inference network model. This paper describes recent improvements to the system as a result of participation in the TIPSTER project and the TREC-2 conference. Improvements include transforming forms-based specifications of information needs into complex structured queries, automatic query expansion, automatic recognition of features in documents, relevance feedback, and simulated document routing. Experiments with one and two gigabyte document collections are also described.

To appear in *Information Processing and Management.*

1 Introduction

The effectiveness of an information retrieval (IR) system depends upon representation and matching. The system must *represent* the information need, it must *represent* the documents, and it must determine how well the information need *matches* each document. Our approach has been to use improved representations of document text and queries in the framework of the inference network model of retrieval. This model uses Bayesian networks to describe how text and queries should be used to identify relevant documents [11; 6; 12]. Document retrieval and routing are viewed as probabilistic inference processes that compare text representations based on different forms of linguistic and statistical evidence to representations of information needs based on similar evidence from natural language queries and user interaction. Learning techniques are used to modify the initial queries both for short-term and long-term information needs (relevance feedback and routing, respectively).

This approach, generally known as the inference net model and implemented in the INQUERY system [4], emphasizes retrieval based on combination of evidence. Different text representations (such as words, phrases, paragraphs, or manually assigned keywords) and different versions of the query (such as natural language and Boolean) can be combined in a consistent probabilistic framework. This type of "data fusion" has been known to be effective in the information retrieval context for a number of years, and was one of the primary motivations for developing the inference net approach.

Another characteristic of the inference net approach is the ability to capture complex structure in the network representing the information need (i.e. the query). A practical consequence of this is that complex Boolean queries can be evaluated as easily as natural language queries to produce ranked output. It is also possible to represent "rule-based" or "concept-based" queries in the same probabilistic framework. This has led to us concentrating on automatic analysis of queries and techniques for enhancing queries rather than on in-depth analysis of the documents in the database. In general, it is more effective (as well as efficient) to analyze short query texts rather than millions of document texts. The results of the query analysis are represented in the INQUERY query language which contains a number of operators, such as #SUM, #AND, #OR, #NOT, #PHRASE, and #SYN [13; 4]. These operators implement different methods of combining evidence.

Some of the specific research issues we are addressing are morphological analysis, the use of phrases and other syntactic structure, the use of feature recognizers (for example, company and country name recognizers) in representing documents and queries, analyzing natural language queries to build structured representations of information needs, learning techniques appropriate for routing and structured queries, techniques for acquiring domain knowledge by corpus analysis, and probability estimation techniques for indexing.

The TIPSTER and TREC evaluations have made it clear that much remains to be learned about retrieval and routing in large, full-text databases based on complex information needs. On the other hand, we have made considerable progress in developing effective techniques for this environment, and the evaluations have shown that good levels of performance can be achieved.

> John Davenport, 52 years old, was appointed chief executive officer of this international telecommunications concern's U.S. subsidiary, Cable & Wireless North America Inc. Mr. Davenport, who succeeds John Zrno, is currently general manager of the group's operations in Bermuda.

Figure 1: Indexing example: Original document text.

> john davenport 52 year old appoint chief execut offic intern telecommun concern u.s. #USA subsidiar cabl wireless north america inc #COMPANY davenport succee john zrno current gener manag group oper bermuda #FOREIGNCOUNTRY

Figure 2: Indexing example: Document text indexed.

2 The INQUERY System

The INQUERY document retrieval and routing system is based on the inference network model [13; 4]. The main processes in INQUERY are document indexing, query processing, query expansion, query evaluation, and relevance feedback. Each is described below.

2.1 Document Indexing

The *document parser* is a set of text processing modules, organized into four phases: (1) layout analysis, (2) lexical analysis, (3) syntactic analysis, and (4) feature recognition. *Layout analysis* is the only phase in which the document can be modified. It transforms the raw document into a canonical format, saving structural information as necessary, and identifies which portions to index. *Syntactic analysis* verifies that the document conforms to an expected format. The other two phases, *lexical analysis* and *feature recognition*, record the locations of terms (words or numbers) and features (company names, countries, etc) in the document text.

The *lexical analysis* module identifies and records word boundaries, recognizes *stopwords*, *stems* the words, and indexes the words for retrieval. In theory, every word in the document collection will be indexed. In practice, it is helpful to identify very common words, such as *operators* or *closed-class* words, which do not carry any meaningful information for retrieval purposes (although they may offer significant information for text extraction [10]). These *stopwords* are usually not indexed, although they are retained in the text so that subsequent textual analysis (syntactic analysis, feature recognition) may make use of them. Stopwords can be indexed, however, if they are capitalized (but not at the start of sentences) or joined with other words (e.g. "the The-1 system"). Stemming is performed to conflate words that have the same root form or stem, in spite of different endings.

Feature recognition is an important step in representing text at different levels of abstraction. Feature recognizers search text for words that correspond to simple semantic components, for example company names or country names. The document is indexed by both the words (e.g. "Lotus Development Corp") and the feature (e.g. #company). The set of feature recognizers delivered with INQUERY is shown below.

Company name recognizer: For each mention of a company in the text, generates a

> Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for the chemicals. pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales

Figure 3: Natural language query text.

> market strateg carr #usa compan #company agricultur chemic report predict market share chemic report market statist market agrochem #usa pesticid herbicid fungicid insecticid fertil predict sale stimul demand price cut volum sale

Figure 4: Query text, after stopword and stop-phrase removal.

transaction for the special term #COMPANY.

U.S. city recognizer: For each mention of a U. S. city in the text, generates a transaction for #CITY.

Country recognizer: For each mention of a country in the text, generates a transaction for either #USA or #FOREIGNCOUNTRY.

These *features* extend the range of queries that can be specified. Figures 1 and 2 illustrate the role that these features play in document indexing. This completes the usual processing for document text.

The INQUERY text processing behavior is customized easily. We have discussed the default behavior, but these modules can be replaced easily if some other behavior is desired.

The document indexing process also involves building the compressed inverted files that are necessary for efficient performance with very large databases. Since positional information is stored, the indices are typically about 40% of the size of the original document collection, after compression.

2.2 Query Processing

Queries can be made to INQUERY by using either natural language or a structured query language or a mixture of the two. Natural language queries are transformed incrementally into complex structured queries in the INQUERY query language by a series of query text processing modules [3]. Query text processing must minimally mirror the indexing text processing. But because query texts are much shorter than document collections, it is practical to experiment with more thorough textual analysis at the research and development stage. This reduces the need to repeatedly index large document collections in order to make small experimental adjustments. All query text processing is experimental and the sequence of operations is adjusted frequently as more is learned about the effects of this processing.

Currently, INQUERY has a small number of internal query text processors [3]. These include stop-phrase removal (e.g., "A document must discuss"), conversion of hyphenation and sequences of capitalized words (proper names) into proximity constraints, insertion of features, case conversion, stopword removal and stemming (Figures 3 and 4).

```
<DOC>
<TITLE> amnesty program </TITLE>
<TEXT>
... 1986(3), act(3), control(2), immigrant/immigration(16), law(8), reform(4), 1982(2),
1987(3), agency(3), aliens(13), duarte(2), el-salvador(2), employers(8), documenta-
tion(2), guatemala(2), file(3), government(6), fear(2), entered(3), illegal(14), natural-
ization(2), mandates(2), legalization(3), nelson(5), nicaragua(2), new-york(2), perma-
nent(4) ...
</TEXT>
</DOC>
```

Figure 5: A PhraseFinder pseudo-document for the concept *amnesty program*. Numbers in parenthesis indicate the number of repetitions of the word preceding.

Orthographic clues such as hyphenation and capitalization, when reliable, are very good clues to phrasal grouping. Hyphens are generally discarded during indexing so that expressions such as *Iran-Contra* or *voice-activated* are indexed as terms *Iran* and *Contra* or *voice* and *activated*, respectively. In query processing, the corresponding procedure is to remove the hyphen and to place a proximity constraint on the words, as shown below.

`voice-activated` ⇒ `#1(voice activated)`

Groups of capitalized words are similarly constrained, as shown below.

`House of Representatives` ⇒ `#1(House of Representatives)`

2.3 Query Expansion

Our approach to query expansion (called *PhraseFinder*[1]) is based on the assumption that concepts found in similar lexical contexts may also be related semantically. For example, the words "connectionist" and "neural networks" might occur in similar lexical contexts, but rarely in the same documents. The semantic relationship captured is not necessarily synonymy, because PhraseFinder also might relate "connectionist" and "back propagation", which co-occur but have different meanings.

A PhraseFinder database is an INQUERY database of *pseudo-documents*. Each pseudo-document represents a *concept*, in this case a noun sequence, that occurs in the document collection. The "text" of the pseudo-document consists of words that occur near the concept in the document collection. For example, a PhraseFinder document for a Wall Street Journal collection contains an *amnesty program* pseudo-document that is indexed by *1986*, *act*, *control*, *immigrant*, *law* (Figure 5), ... Although very different in implementation, the approach is similar in spirit to the distributed representations employed by the MatchPlus and Bellcore systems [1].

The usual document retrieval algorithms (discussed in Section 2.4) are used to retrieve the pseudo-documents that represent concepts. Thus, INQUERY can use any structured query to retrieve a ranked list of concepts. The most highly ranked concepts are those that

[1] PhraseFinder was called WordFinder until we discovered that WordFinder is the trademark name of a commercial software product.

Query 115: Impact of the 1986 Immigration Law - will report specific consequence consequences of the U.S.'s Immigration Reform and Control Act of 1986.	Query 132: "Stealth" Aircraft - will provide cost, technical, and/or performance data on U.S. "stealth" aircraft projects.
illegal immigration	northrop corp.
illegals	tactical fighter
undocumented aliens	aerospace companies
amnesty program	flying wing design
immigration reform law	enemy radar
editorial-page article	stealth bomber
naturalization service	development program
civil fines	radar-evading aircraft
new immigration law	bat-winged aircraft
legal immigration	cost overruns
employer sanctions	expensive plane
simpson-mazzoli immigration reform	stealth fighter
statutes	radar-evasion standards
applicability	full-scale production
seeking amnesty	palmdale
legal status	radar-evading
immigration act	pentagon official
undocumented workers	flying wing
guest worker	air force officials
sweeping immigration law	development costs

Figure 6: Query expansion example: Concepts discovered automatically for TIPSTER topics 115 and 132.

are most highly associated (*in that collection*) with the query. Figure 6 shows the top 20 concepts returned by INQUERY for two TIPSTER topics, filtered to remove concepts that already appeared in the query.

A query is expanded by evaluating it against a PhraseFinder database, selecting the top ranked concepts, weighting them, and adding them to the query. Our current approach is to select the top n concepts and weight them at one half the weight of the initial query terms.

PhraseFinder is sensitive to several parameters, including the size of the collocation window, decisions about what syntactic classes to include in the window, and whether low and/or high frequency noun groups are removed from the database. The current implementation of PhraseFinder shows promise on the TIPSTER data, but more work is necessary to build a PhraseFinder that would be effective on a variety of document collections [9].

2.4 Query Evaluation

The query evaluation process uses the inverted files and the query represented as an inference net to produce a document ranking. Documents are ranked according to the belief that they are relevant to the query.

Query evaluation involves probabilistic inference based on the operators defined in the INQUERY query language. These operators define new concepts and how to calculate the belief in those concepts using linguistic and statistical evidence. The belief in a document due to the occurrence of a single term t is:

$$\mathrm{bel}_{\mathrm{term}}(t) \quad = \quad d_b + (1 - d_b) \cdot \left(d_t + (1 - d_t) \cdot \frac{\log(tf + 0.5)}{\log(max\_tf + 1.0)} \right) \cdot \frac{\log(\frac{C}{df})}{\log(C)} \qquad (1)$$

where

tf = the frequency of term t in the document,
max_tf = the frequency of the most frequent term in the document,
df = the number of documents in which term t occurs,
C = the number of documents in the collection,
d_t = minimum term frequency component when a term occurs in a document,
d_b = minimum belief component when a term occurs in a document.

This equation is a variation of the well-known *tf.idf* approach, with values normalized to remain between 0 and 1, and further modified by default term frequency (d_t) and default belief (d_b) values that the user may define at program invocation. d_t and d_b default to 0.4.

The belief in a document due to a given query language operator depends on the type of operator and the belief in its arguments. The query language operators have been discussed in detail elsewhere [13; 4], so we merely provide several examples to illustrate their general operation. #WSUM is a weighted sum operator, #UWn is an unordered window proximity operator, and #PROXn is an ordered interword proximity operator.

$$\mathrm{bel}_{\mathrm{wsum}}(w_1 \cdot Q_1, \ldots, w_m \cdot Q_m) \quad = \quad \frac{(w_1 \cdot \mathrm{bel}(Q_1) + \ldots + w_m \cdot \mathrm{bel}(Q_m)) \cdot w_q}{w_1 + \ldots + w_m} \qquad (2)$$

$$\mathrm{bel}_{\mathrm{prox}n}(Q_1, Q_2, \ldots Q_m) \quad = \quad \mathrm{bel}_{term}(Q') \qquad (3)$$

$$\mathrm{bel}_{\mathrm{UW}n}(Q_1, Q_2, \ldots Q_m) \quad = \quad \mathrm{bel}_{term}(Q'') \qquad (4)$$

where

m = the number of arguments to an operator,
n = a positive integer argument to the proximity operator,
Q_i = a term or a nested query net operator,
w_i = a positive real number, used as a query term weight,
w_q = the maximum value that the #wsum operator can yield (normally 1.0),
Q' = a compound term created temporarily during query evaluation to represent the locations where $Q_i, \ldots, Q_n$ occur in order with an inter-word separation $< n$, and
Q'' = a compound term created temporarily during query evaluation to represent the locations where $Q_1, \ldots, Q_n$ occur in any order within a text window of size $\leq n$.

The efficiency of retrieval is comparable to commercial information retrieval systems.

2.5 Relevance Feedback

The INQUERY system is able to refine queries automatically based upon relevance feedback by a user. The general approach is for the system to select terms from relevant documents, add them to the query, and then reweight all of the query terms.

Early experiments [7] showed that ranking terms by the product of their frequency in relevant documents (*rdf*) and their inverse document frequency (*idf*) was best on small and medium-sized collections with relatively small numbers of relevance judgements. The number of terms added was set empirically to 5. Term weights were determined by their frequency in relevant documents (*rtf*). The INQUERY system still uses this model for interactive relevance feedback, where the number of relevance judgements per query is generally low (e.g. < 15).

3 Use of INQUERY in TIPSTER and TREC

In this section we describe modifications made to INQUERY for the TIPSTER and TREC evaluations. Specifically, we focus on query processing, query evaluation, and relevance feedback/routing.

3.1 Query Processing

TIPSTER query processing is performed as a preprocessing step rather than in the body of the INQUERY program, to simplify research and experimentation. Preprocessing stages are made up of `sed`, `awk`, `flex` (or `lex`) scripts and C code. What follows is a description of the text preprocessing modules that have been used for TIPSTER queries. The order of their use is not fixed, but it can be significant. For example, it was found useful to phrase a hyphenated compound such as *word1-word2* as `#1 (word1 word2)` and to phrase a group of capitalized words as `#3 (word1 word2)`. We have experimented with removing names of countries and some capitalized expressions from medium-sized phrases. For example:

```
#PHRASE (word1 capitalized-group word2) ⇒
    #PHRASE (word1 word2) capitalized-group.
```

It makes a difference whether you process hyphenated and capitalized words before or after you generate the larger #PHRASE.

There are two main kinds of query styles: a natural language query and a keyword or key concept query. For example, the <**desc**> and <**narr**> fields of a TIPSTER topic (Figure 7) represent natural language queries of varying levels of abstraction. The <**con**>, <**title**> and <**fac**> fields represent key concepts in the query. The main difference between the two types of processing is that the key concept query has more controlled information. The phrasing and emphasis are already given and do not have to be conjectured from the language structure. It is valuable to discover how to treat both styles of query, because a good user interface will make it easy for a user to input both styles. For example, a user may enter a prose query and then highlight the important words and phrases in the query in some convenient manner. These highlighted words would then be treated as key concepts in the query processing.

Natural language query fields are tagged for syntactic category by a part-of-speech (POS) tagger [5]. Additionally, we change operator phrases to single words in order to simplify later processing. An example of this simplification is replacing the phrase *in order to* with the infinitive particle *to* or replacing *with respect to* with the word *regarding*. The goal of this

<num> Number: 106
<dom> Domain: Law and Government
<title> Topic: U.S. Control of Insider Trading
<desc> Description:
Document will report proposed or enacted changes to U.S. laws and regulations designed to prevent insider trading.
<narr> Narrative:
A relevant document will contain information on proposed or enacted changes to U.S. laws and regulations, including state laws and stock market rules, which are aimed at increasing penalties or closing loopholes in existing institutional discouragements to insider trading. NOT relevant are reports on specific insider trading cases, such as the prosecutions and settlements related to the Boesky - Milken - Drexel Burnham Lambert scandal, unless the report also contains specific information on legal or regulatory change.
<con> Concept(s):
1. insider trading
2. securities law, bill, legislation, regulation, rule
3. Insider Trading Sanctions Act, Insider Trading and Securities Fraud Enforcement Act
4. Securities and Exchange Commission, SEC, Commodity Futures Trading Commission, CFTC, National Association of Securities Dealers, NASD
<fac> Factor(s):
<nat> Nationality: U.S.

Figure 7: Query processing example: Original query.

```
#WSUM ( 1.0
!Terms from <title> field:
2.0 #UW50 (Control of Insider Trading)
2.0 #PHRASE (#USA Control) 5.0 #PHRASE (Insider Trading)
! Terms from <con> field:
2.0 #PHRASE (securities law) 2.0 bill 2.0 legislation 2.0 regulation 2.0 rule
2.0 #3 (Insider Trading Sanctions Act)
2.0 #3 (Insider Trading and Securities Fraud Enforcement Act)
2.0 #3 (Securities and Exchange Commission) 2.0 SEC
2.0 #3 (Commodity Futures Trading Commission) 2.0 CFTC
2.0 #3 (National Association of Securities Dealers) 2.0 NASD
! Terms from <desc> field:
1.0 proposed 1.0 enacted 1.0 changes 1.0 #PHRASE (#USA laws)
1.0 regulations 1.0 designed 1.0 prevent
! Terms from <fac> field:
2.0 #NOT(#FOREIGNCOUNTRY) )
```

Figure 8: Query processing example: Automatically processed query.

replacement is to remove phrases that resemble noun phrases syntactically but that are really syntactic operators (e.g., phrasal prepositions) with no substantive content.

When the text is tagged and the potentially irrelevant material removed, syntactically-based noun group capture is performed. Certain kinds of noun phrase patterns are enfolded in a #PHRASE operator (Figure 8):

1. A noun phrase that contains more than one modifying adjective and noun is enclosed in a #PHRASE operator;

2. A head noun with no premodifiers and followed by a prepositional phrase is enclosed in a #PHRASE operator with the head noun of the prepositional phrase;

All text in the query is searched for constraint expressions. Among these expressions are "company", "not U. S." or a restriction in the nationality section of the <**fac**> field to the U.S. or another nation. A restriction to U.S. nationality as the area of interest is implemented by penalizing documents for references to foreign countries. A restriction to other nationalities is implemented by repeating that country as a term. This asymmetry depends on the fact that the document collection is drawn solely from U.S. sources, and therefore the U.S., as the default area of interest, is rarely referred to unless a government body or foreign policy implementation is under discussion (Figure 8).

There is some recognition of simple time expressions, such as "since 1984," which are expanded to the set of years that might be intended by the phrase in question.

Countries are recognized as such and are handled so that expressions like "South Africa" are phrased as `#1(south africa)` even when they appear in the middle of a larger group of capitalized words. Proper names such as country names are moved out of the scope of #PHRASE operators, since it generally increases the effectiveness of a #PHRASE to reduce the number of words in it. Nationality constraints can better be maintained within the scope of the larger and more tolerant #SUM operator. For example the phrase

"import ban on South African diamonds"

becomes by stages,

```
#PHRASE (import ban on #SYN (#1 (south african) #1 (south africa))
            diamonds)
```

and finally

```
#SUM (#SYN (#1(south african) #1(south africa))
        #PHRASE(import ban on diamonds)).
```

Key concept query processing is different from prose query processing since the concept separation provided by the user can presumably be trusted. Instead of using a part-of-speech tagger, we rely on comma delimitation of concepts, and #PHRASE the words found between each pair of delimiters (Figure 8: Terms from <con> field).

Additionally, if any constraints were found anywhere else in the query, e.g., a mention of the word "company" or an exclusionary geographical constraint (e.g., "not USA" or "only USA"), the query will be modified according to these constraints. For example (Figure 8),

"only USA" ⇒ **#NOT (#FOREIGNCOUNTRY)**

and

"not USA" ⇒ **#NOT (#USA)**.

If the word "company" is found in a query, then a second copy of the key concepts (the <**con**> field), is produced where each item in the field appears in an unordered window operator with the feature #COMPANY. For example, if "South Africa" appears as a key concept and "company" appears somewhere in the query), then the preprocessor produces the query text **#UW50 (#COMPANY #1 (south africa))**, which matches any document that has a company name within fifty words of "South Africa".

Term weights depend upon the TIPSTER topic field in which the term occurred. Terms from the Title and Concept fields get twice the weight of terms from the Description and Narrative fields. Simple query optimization may further alter term weights by collapsing multiple occurrences of a query term into one occurrence carrying the sum of the weights (e.g. #PHRASE (Insider Trading) in Figure 8).

We have experimented with manual modification of processed queries in order to measure the feasibility and effectiveness of simple user adjustments to automatic query processing output. We have explored simple modifications such as adding a term from the Narrative field, deleting a term, and constraining existing terms to appear near each other in a document (Figure 9). This has sometimes proved to be effective in increasing the quality of retrieval results.

3.2 Query Evaluation

The formula that determines belief in a document due to the occurrence of a term (Equation 1) scales the log of tf by the log of max_tf, producing values in the range $[0, 1]$. One consequence of this approach is that it favors long documents. For example, if a term occurs 3 times in a document and the most frequent term occurs 6 times, the result is $\frac{\log(3+0.5)}{\log(6+1)} = 0.644$. However, if the term and the most frequent term both occur 10 times

```
#WSUM (1.0
2.0 #UW50 (Control of Insider Trading)
3.0 #3 (Insider Trading) 1.0 #3 (securities law)
2.0 #uw50 (#syn (bill law regulation rules) insider trading)
1.0 #3 (Insider Trading Sanctions Act)
1.0 #3 (Insider Trading and Securities Fraud Enforcement Act)
1.0 #3 (Securities and Exchange Commission) 1.0 SEC
1.0 #3 (Commodity Futures Trading Commission) 1.0 CFTC
1.0 #3 (National Association of Securities Dealers) 1.0 NASD
2.0 #NOT (#FOREIGNCOUNTRY)
! Terms extracted manually from the Narrative:
1.0 #3 (increasing penalties) 1.0 #3 (closing loopholes)
1.0 #NOT (Boesky) 1.0 #NOT (Milken)
1.0 #NOT (#3 (Drexel Burnham Lambert)))
```

Figure 9: Query processing example: Manually modified query.

more often in another document, the result is $\frac{\log(30+0.5)}{\log(60+1)} = 0.831$.

A slight bias towards long documents is reasonable, so that the system is not unduly influenced by the occurrence of a single term in a short document. However, this bias proved too strong for a collection in which document lengths varied greatly.

A quick solution was to include a bias against very long documents. The method chosen (Equation 5) reduced the default tf value for long documents.

$$\begin{aligned} \mathrm{bel}_{\mathrm{term}}(Q) &= d_b + (1-d_b)\cdot\left(d_t \cdot H + (1-d_t)\cdot\frac{\log(tf+0.5)}{\log(max\_tf+1.0)}\right)\cdot\frac{\log(\frac{C}{df})}{\log(C)} \qquad (5) \\ H &= \begin{cases} 1.0 & \text{if } max\_tf \leq 200 \\ \frac{200}{max\_tf} & \text{otherwise} \end{cases} \end{aligned}$$

The penalty H was effective, if not pleasing theoretically. It prevented INQUERY from being biased unduly towards long documents, but still allowed them to be retrieved. Only four of the TREC-2 systems retrieved more relevant Federal Register documents than did INQUERY [8]. Each of those systems also retrieved at least twice as many non-relevant Federal Register documents as did INQUERY.

3.3 Routing

Our approach to the *routing* portion of our TIPSTER and TREC work was based initially upon our existing relevance feedback mechanisms. Routing profiles were constructed by a two step process. The first step was to produce automatically a query representing each TIPSTER topic, as described above in Section 2.2. The second step was to modify the query, using relevance feedback. This modified query was then used as a routing profile in the routing experiments.

Experiments with creating routing profiles showed that better results could be obtained by replacing the *idf* component of the term selection algorithm with $\log\frac{df}{tf}$, where *df* is the number of documents in which the term occurs (*document frequency*) and tf is the frequency of the term in the collection. The number of terms added to the query was also increased, from 5 to 30. This modified algorithm appears effective even with small numbers of relevance judgements.

The addition of proximity operators further improves the average precision of routing profiles. INQUERY considers every pair of terms within a distance of n in a relevant document as a potential source of a proximity operator to add to a query. Experiments with values of n ranging from 3 to 50 showed that a range of values is superior to any single value. The resulting set of pairs, which can be quite large, is filtered to remove pairs that occur rarely in relevant documents. The resulting pairs are ranked by the formula below.

$$\left(\frac{rdf}{|R|} - \frac{ndf}{|NR|}\right) \cdot rtf$$

$|R|$ is the number of relevant documents, rdf is the number of relevant documents in which the pair co-occur, rtf is the number of times the pair co-occur in relevant documents, ndf is the number of non-relevant documents in which the pair co-occur, and $|NR|$ is the number of non-relevant documents. In our TIPSTER experiments, 10 unordered window proximity (#UWn)[2] operators with $n = 5$, and 20 #UWn operators with $n = 50$, were added to each query. These operators were intended to capture phrase-level and paragraph-level co-occurrence.

The TIPSTER document collection differs from previously available document collections in that it contains many more documents and many more relevance judgements per query. One might expect having more relevance judgements to improve the reliability of the statistics obtained by analyzing relevant documents, but it is not clear that this is so. Experiments showed that INQUERY's performance improved steadily as the number of relevant documents used was increased to about 275-300 documents. After 300 relevant documents, performance began to degrade slowly. Further work is required to understand this behavior.

It can be argued that several hundred relevant documents are a better representation of a user's interest than the query that retrieved them along with irrelevant documents. We found that better results were obtained by discarding the user's original query and creating a completely new routing query using the relevance feedback methods described above. Figure 10 shows a query created by this method.

In addition to the number of relevance judgements, it is unusual to have relevance judgements from a diverse set of systems. In an operational setting, even over long term use, one is likely to only have relevance judgements resulting from use with a single system. We found that restricting INQUERY's attention to only those relevant documents that it retrieved *reduced* the number of relevance judgements needed to reach a given level of performance. Using relevant documents retrieved by many systems (e.g. the TREC–1 systems) eventually yielded similar performance, but required analysis of many more relevant documents.

The routing experiments show that it is feasible to automatically construct relatively accurate profiles in an operational setting. Profiles can be created from a set of relevant documents, or from repeated interaction with a user. Either approach will yield relatively accurate routing profiles. The experiments also showed that, even when large numbers of

[2] The #UWn operator looks for co-occurrence in any order in a text window of size n.

```
#q051 =
#WSUM( 1.000000 .433963 dougla 30.622835 subsid 14.105722 mcdonnel 2.856207 spain
22.664160 boe 30.620134 european 5.776313 g.m.b.h. 8.629494 340 14.828697 messerschmit
24.899202 industri 7.524240 jet 28.518532 aerospac 6.187950 unfair 34.157051 aircraft
5.245394 construccion 5.942457 330 12.249618 boelkow 5.435017 west 5.136472 franc
8.268916 aerospatial 5.439325 aeronautica 6.971968 jetlin 11.957228 blohm 9.611669 german
10.252533 mbb 25.656782 consortium 16.704779 british 138.805618 airbu 10.874762
plane 2.533194 plc 2.73149 #UW5( #company #foreigncountry ) 6.74627 #UW50( 330
airbu ) 6.36442 #UW50( aid airbu ) 6.03555 #UW50( airbu messerschmit ) 8.87131
#UW50( aircraft subsid ) 7.39724 #UW5( british aerospac ) 11.1438 #UW50( british
airbu ) 3.45497 #UW50( competitor airbu ) 6.27218 #UW50( cost airbu ) 20.7534
#UW50( european airbu ) 4.8756 #UW50( g.m.b.h. airbu ) 14.6286 #UW50( german
airbu ) 23.6137 #UW50( govern airbu ) 4.41921 #UW5( govern european ) p3.63681
#UW50( help airbu ) 4.04575 #UW5( mcdonnel boe ) 8.33751 #UW5( mcdonnel dougla
) 3.19623 #UW5( offic u.s. ) 8.1083 #UW50( partner airbu ) 4.9825 #UW50( price airbu
) 6.19649 #UW50( project airbu ) 10.3209 #UW50( say airbu ) 18.1742 #UW50( subsid
airbu ) 15.8317 #UW50( trade airbu ) 25.5183 #UW50( u.s. airbu ) 5.23789 #UW5(
u.s. trade ) 2.04795 #UW5( wall street ) 11.3886 #UW50( west airbu ) 6.19697 #UW5(
west german ) )
```

Figure 10: Routing profile created automatically from relevant documents.

relevant documents are available for analysis, a combination of automatic query formation and manual query construction (from previous ad hoc experiments) is superior to either approach alone.

4 Experiments and Results

INQUERY was evaluated as part of TREC and as part of the TIPSTER program. TREC is helpful because it evaluates systems on a broad set of relevance judgements collected by a variety of information retrieval systems. However, the number of experiments one can perform in TREC is limited. TIPSTER evaluated systems on a more narrow set of relevance judgements collected by three systems, but a larger number of experiments was permitted. In this section, we describe results of both TREC and TIPSTER evaluations.

Unless otherwise noted, the results shown are average precision over 11 recall points, based upon a full-ranking of documents, using the TREC-2 relevance judgements. This methodology makes the TIPSTER and TREC results directly comparable.

4.1 The TREC Experiments

Four experiments were submitted to the TREC evaluation, two “ad-hoc” and two “routing”. In these experiments, we emphasized automatic query processing and automatic feedback algorithms for routing. The following is a summary:

- AdHoc: topics 101-150 against TIPSTER volumes 1 and 2.

INQ001: Created automatically from TIPSTER topics. Contains phrases. Details of query processing used are described above.

INQ002: INQ001 queries, modified manually. Modifications were restricted to eliminating words and phrases, and adding paragraph-level operators around existing words and phrases. The method was somewhat different than the method used at last year's TREC conference, as discussed below.

- Routing: topics 51-100 against TIPSTER volume 3.

INQ003: Created automatically from TIPSTER topics and relevance judgements from Volumes 1 and 2. Baseline queries (from a previous TIPSTER evaluation) were modified by reweighting and adding single-word terms. The term weighting and selection function used was df.idf, as described in [7]. Only the top 120 relevant documents found by INQUERY were used for feedback, and 30 terms were added to each query.

INQ004: Formed by combining (using the #SUM operator) INQ003 queries and INQRYP queries (used in TIPSTER 18 month evaluation). The INQRYP queries were produced automatically and then modified manually. Modifications were restricted to eliminating words and phrases, and adding paragraph-level operators around existing words and phrases.

Table 1 gives the results for the adhoc queries. There was little difference in effectiveness between the automatically processed queries and the semi-automatically processed queries. This result is surprising given the large difference we observed in the previous TREC. One reason for this difference is that query processing for the automatically processed queries has been significantly improved, as described in the previous section. Another reason is that this time paragraph-level concepts were formed in a much more mechanistic way and were constrained by the language of the Description and Narrative fields. In the previous conference, the only constraint was the vocabulary used in the queries, and the user's "world knowledge" was used to group concepts. The earlier approach resulted in considerably better retrieval performance. Additional experiments using manually edited queries are discussed in the next section.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ001 (automatic baseline):	.62	.58	.49	.39
INQ002 (manual, simulated NLP):	.60 (−3.9%)	.59 (+2.1%)	.51 (+2.6%)	.39 (−0.4%)

Table 1: Results for Adhoc queries

The routing results (Table 2) show that some improvement is obtained by combining the manual queries with the queries that were automatically modified using relevance feedback techniques. The difference in performance between the two types of queries is considerably less than last year, however. Our own experiments have also shown that no additional gains in performance were obtained by using more than the top 150 documents from the INQUERY output. This is a significant result from a practical viewpoint, since in an operational environment we will not want to rely on having output from other systems or need thousands of relevance judgements before performance improves.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ003 (relevance feedback):	.65	.55	.44	.38
INQ004 (`#SUM (INQ003 INQRYP)`):	.66 (+1.8%)	.58 (+4.7%)	.45 (1.4%)	.39 (+3.2%)

Table 2: Results for Routing queries

4.2 The TIPSTER Experiments

In the TIPSTER 24 month evaluation, which took place soon after the TREC-2 evaluation, we did a number of experiments that complement those done in TREC. In particular, we evaluated paragraph-based retrieval, expansion using an automatically generated thesaurus, and feedback techniques that use phrases. In this section, we report some of the most interesting results. The precision figures given here are calculated using the TREC-2 relevance judgements, rather than the TIPSTER judgements.

Table 3 shows the results of the ad-hoc runs. INQ009 is the baseline result obtained using automatic query processing on the TIPSTER topics, excluding the Narrative field.[3] INQ010 and INQ041 are the results obtained by manually modifying the queries produced for INQ009. In the case of INQ010, the queries were modified by adding natural language structures from the Narrative that a sophisticated parser with limited lexical semantics might reasonably be expected to extract. For INQ041, in making modifications, the user was allowed to use world knowledge when deciding what to delete, reweight, structure or extract from the Narrative. This latter approach was effective in the first TIPSTER and TREC evaluations.

These results show that manually modified queries can achieve significantly better precision at low recall levels. For example, at the 5 document cutoff level, the average precision for INQ041 is 10.3% higher than INQ009. The overall average is similar, however. This is a much smaller difference than was seen in the first TREC and TIPSTER evaluations of the INQUERY system. The result may be due to the fact that the automatic query processing has improved considerably, or it may be due to the difficulty of the topics in the third set.

[3] The National Institute of Standards and Technology required that the TREC and TIPSTER query sets have different identifiers, even if the query sets were identical. The INQ009 and INQ001 query sets differ on one query, due to a minor error being fixed between the TREC-2 and TIPSTER 24 month evaluations. The INQ010 and INQ002 query sets are identical.

[4] The precision figures shown for INQ011 and INQ012 were supplied by NIST, as part of the TIPSTER evaluation. They are based on TIPSTER relevance judgements. NIST rated the average precision (non-interpolated) over all relevant documents of INQ011 as 27.9% worse than INQ009, and INQ012 as 4.3% better than INQ009. These figures, while interesting, are not directly comparable to the 11 point average precision shown for other experiments. Our approach to paragraphs had changed sufficiently by the time TREC relevance judgements became available that we could not easily generate full recall/precision tables for INQ011 and INQ012.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ009 (baseline):	.62	.58	.49	.39
INQ010 (manual, simulated NLP):	.60 (−3.9%)	.59 (+2.1%)	.51 (+2.6%)	.39 (−0.4%)
INQ015 (PhraseFinder 1):	.60 (−3.2%)	.59 (+1.9%)	.50 (+0.8%)	.39 (+1.2%)
INQ016 (PhraseFinder 2):	.60 (−3.2%)	.59 (+1.9%)	.50 (+0.8%)	.39 (+0.9%)
INQ041 (manual):	.68 (+10.3%)	.60 (+4.5%)	.50 (+0.4%)	.39 (+0.8%)
INQ044 (`#SUM (INQ009 INQ041)`):	.65 (+4.5%)	.61 (+6.4%)	.51 (+3.5%)	.41 (+6.6%)
INQ011 (Paragraph):	.45 (−27.7%)	.45 (−21.5%)	.41 (−17.5%)	N/A[4]
INQ012 (Doc+Par/2):	.64 (+2.6%)	.57 (−0.2%)	.51 (+2.5%)	N/A[4]

Table 3: Ad-hoc results.

The results for INQ010 also suggest that using NLP techniques to analyze the Narrative section of a topic may not improve the query.

The INQ015 and INQ016 results were for an early version of the PhraseFinder query expansion system. Although these results show no significant differences, better PhraseFinder results are presented below.

The automatically processed queries and the manually modified queries are two different representations of the information need. Experience has shown that combining different sources of evidence can yield superior results. One experiment combined these two versions of the information need using the INQUERY framework. The result of this combination (INQ044) was slightly worse than INQ041 at the 5 document cutoff level, but overall was better than either the automatic or manual queries on their own (Table 3).

The INQ011 experiment investigated using the query to rank document paragraphs and then assigning each document the score of its best matching paragraph. Paragraph boundaries are not marked in this collection, so they were inferred from indentation and other orthographic clues. The INQ012 experiment combined the results of document-level representations with paragraph-level representations, using the INQUERY #WSUM operator. An improvement in performance was obtained when the paragraph-level results were weighted at 1/2 the importance of the document-level results. The performance of the paragraph-level search on its own was poor, because paragraph boundaries in this collection often do not indicate content shift in documents.

Table 4 shows the results of the routing experiments. INQ026 is the result of using the automatically processed version of the original queries with no relevance feedback. INQ020 is the result of using simple techniques for reweighting terms and adding thirty new terms based on feedback from relevant documents in the earlier databases. It can be seen that these feedback techniques result in significant improvements.

INQ022 shows the result of using the manually modified version of the query (no relevance feedback), and INQ021 gives the combination of the manual queries and the queries produced using simple relevance feedback. Once again the combination results in an improvement, although it is small for this experiment due to the relatively poor performance of the manual queries.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ026 (automatic baseline):	.59	.51	.39	.34
INQ020 (relevance feedback):	.66 (+12.9%)	.57 (+11.0%)	.45 (+15.0%)	.39 (+16.7%)
INQ022 (INQ020 w/user:)	.63 (+7.5%)	.53 (+4.3%)	.41 (+4.1%)	.34 (+2.7%)
INQ021 (`#SUM(INQ020 INQ022)`):	.70 (+18.4%)	.58 (+12.9%)	.46 (+16.0%)	.40 (+19.0%)
INQ023 (INQ020 w/prox):	.67 (+13.6%)	.60 (+16.8%)	.47 (+19.8%)	.41 (+22.3%)
INQ024 (INQ020 w/o INQ026):	.68 (+15.7%)	.59 (+16.1%)	.46 (+16.5%)	.42 (+24.2%)

Table 4: Routing results.

INQ023 and INQ024 show the result of using more complex relevance feedback techniques in which proximity structures (paragraph and phrase level) were extracted from relevant documents as well as simple terms. Twenty paragraph-level proximities, ten phrase-level proximities and thirty terms were added. A phrase-level concept is a #UW5 two-word pattern that occurs frequently in the relevant documents, and a paragraph-level concept is a #UW50 two-word pattern. Both phrase-level and paragraph-level proximities produced significant improvements. The best result (INQ024) was from a run where the original query was ignored and all terms came from relevant documents.

These results show that there is little difference between using the original query or just the relevant documents. This is probably due to the large number of relevance judgements available in this routing experiment. In a relevance feedback situation, where there are far fewer relevant documents, the original query is very important. It is clear that the addition of phrase and paragraph-level structure to the routing query has improved performance. The average precision for INQ023 is 4.8% higher than INQ020. Combining these new runs with manually modified routing queries produced further improvements.

Table 5 shows additional results using PhraseFinder. TipC and TipT were the results of using topics 51-100 expanded using the best 5 unique concepts, and an average of 6 duplicate concepts, retrieved by PhraseFinder.[5] These queries were run against the third TIPSTER disk. For TipC, the query used to search PhraseFinder was the Concepts field from each topic, whereas for TipT, it was the Topic field. The results show substantial improvements. They were all obtained using a training set of 250,000 documents from WSJ, AP and Ziff to build the PhraseFinder database. The results in S3T, T3C, T5C and T10C were obtained using a smaller 50,000 document collection as the basis for PhraseFinder. The results, although not as good as with the larger database, are still significant. The T3C, T5C and T10C experiments show the effects of changing the size of the collocation window used to index noun groups.

[5] If a PhraseFinder concept was already in the query, it is called a ***duplicate***, otherwise it is ***unique***. All concepts ranked (by PhraseFinder) above the fifth unique concept were added to the query.

Query Type	**Average Precision**			
	5 Docs	**30 Docs**	**100 Docs**	**11-Pt Avg**
INQ026 (automatic baseline):	.59	.51	.39	.34
TipC (Concept fld, 250K db, 5 win):	.64 (+8.2%)	.54 (+4.9%)	.42 (+5.3%)	.36 (+6.5%)
TipT (Topic fld, 250K db, 5 win):	.58 (−0.7%)	.53 (+3.3%)	.40 (+2.3%)	.34 (+2.4%)
S3T (Topic fld, 50K db, 3 win):	.58 (−2.0%)	.53 (+3.5%)	.40 (+1.8%)	.34 (+2.1%)
T3C (Concept fld, 50K db, 3 win):	.61 (+4.1%)	.53 (+3.9%)	.41 (+4.3%)	.35 (+4.7%)
T5C (Concept fld, 50K db, 5 win):	.60 (+2.0%)	.52 (+1.4%)	.41 (+3.1%)	.35 (+4.4%)
T10C (Concept fld, 50K db, 10 win):	.62 (+4.8%)	.53 (+3.9%)	.42 (+6.4%)	.36 (+6.2%)

Table 5: PhraseFinder results. INQ026 is the baseline result. TipC through T10C show the effects on INQ026 of query expansion under varying conditions.

5 Efficiency Issues

INQUERY was developed to run under the UNIX operating system, on workstations manufactured by Digital Equipment Corporation, and SUN Microsystems. It has been ported to the MS-DOS operating system (with and without the Windows graphical user interface) on personal computers containing the Intel 486 microprocessor. These hardware platforms include 16, 32 and 64 bit architectures.

The amount of memory and disk space required for depends on the size of the document collection. For a collection of N bytes, INQUERY needs about $5N$ bytes of disk space to build its document database. Once the database is built, INQUERY needs about $1.5N$ bytes of disk space to store the document database (N bytes for the raw text, $0.5 \cdot N$ bytes for the indices). Memory requirements are more difficult to predict, because they depend upon the characteristics of the document collection, and the complexity of the queries. For UNIX workstations, a very rough estimate is that INQUERY requires about $\frac{N}{15}$ bytes of virtual memory for TIPSTER queries. A reasonable amount of physical memory is $\frac{N}{60}$. Therefore a 2 gigabyte collection would need a minimum of about 135 MB of virtual memory and 32 MB of physical memory. For PCs running DOS, about $\frac{N}{15}$ bytes of physical memory is needed.

Although INQUERY's appetite for memory and disk space is not unreasonable when compared with comparable information retrieval systems, it is being reduced. Experiments have been conducted with an in-memory approach to document indexing that significantly reduces the disk space needed during index creation. The advantages of this approach are its simplicity for the user, and a reduction of the peak disk space usage from about $5N$ bytes to $1.9N$ bytes. The disadvantage is that permanent disk usage is increased from $1.5N$ bytes to $1.9N$ bytes. Experiments are also being conducted with a different approach to document retrieval that will allow a user or system administrator to control the amount of memory consumed by INQUERY, essentially trading memory for response time.

The INQUERY system builds document collections automatically at about 40–50 megabytes per CPU hour on a SUN SPARCserver 690 UNIX system with 128 MB of physical memory. Speed varies with the size of the document collection, because transaction sorting takes time proportional to $n \log n$.

On the same UNIX system, document retrieval takes an average of about 1 CPU second per query term on a 1 gigabyte document collection. The time varies widely, depending upon the frequency of the term in the collection and the type of query language operators used. Proximity and synonym operators require considerably more time and space than do operators like #WSUM, #AND and #NOT that ignore locations of terms in a document.

6 Conclusions

The TIPSTER and TREC evaluations have demonstrated that the INQUERY approach to retrieval and routing is both effective and efficient. We have shown that the probabilistic framework is portable, trainable and improvable. The extensibility and robustness of this approach are further demonstrated in technology transfer efforts involving INQUERY. Apart from these general accomplishments, however, we can be more specific about the lessons that have been learned in the major areas of work.

The most important lesson was that sophisticated query processing produces significant improvements. We developed a variety of query processing techniques that together improved the overall system effectiveness considerably. In general, automatically processed queries were competitive with hand-crafted queries.

We also learned that highly structured routing profiles created automatically from relevance judgements consistently outperform profiles created semi-automatically and manually. One of the most effective techniques is the automatic inclusion of proximity pairs in the profile.

The shift from indexing a static set of phrases during document indexing to dynamic extraction of phrases during query processing revealed the difficulty of finding evidence for phrases in documents. Straightforward methods appear effective for recognizing phrases in queries, but it remains unclear how best to recognize when a phrase matches a document.

The TIPSTER topics are much longer than typical IR queries. The mutual disambiguation produced by the presence of so many terms makes word sense disambiguation a marginal technique. For the same reason, simple query expansion techniques, such as using a general thesaurus, were not effective in this environment. However, PhraseFinder demonstrates that more sophisticated automatic query expansion can still yield significant improvements.

The idea of combining multiple sources of evidence turned out to be central to our work. For example, we showed that paragraph-level matching can produce significant improvements in effectiveness when combined (in the INQUERY framework) with document-level matching. We also showed that manually-modified queries can improve results when combined with automatically processed queries.

Feature extraction/recognition appears to be most effective in narrow domains. Our experiments with including extraction in the indexing and retrieval process showed only small effectiveness improvements in TIPSTER. Our experience with other collections have shown more promise.

7 Future Work

Work with the TIPSTER/TREC collection raises as many questions as it answers. In this section, we focus on four of the most interesting areas for future research with INQUERY: the Narrative field, estimation, PhraseFinder, and passage retrieval.

The Narrative field of the TIPSTER topic describes precisely the criteria that make a document relevant. It would seem to make sense to incorporate those criteria into the query. However, we found it safer to ignore the Narrative than to use it.

The Narrative is difficult for at least two reasons. First, it describes subjects that are *required*, *desirable*, and *prohibited* in relevant documents. Distinguishing among these categories during query processing can be difficult. Second, the Narrative describes the subjects at a different level of abstraction than is used in documents. The IR system must figure out that "changes to U.S. laws" in the Narrative matches "amendment passed" in a document text.

Further improvements in precision and recall are possible if the estimation formulae (e.g. Equation 1) and stopword list are *tuned* for TIPSTER/TREC. We avoid tuning for a collection by requiring that any change maintain or improve results on several test collections. However, the fact that tuning is effective suggests that further improvements to INQUERY are possible.

One priority is removal of max_tf, in order to produce more stable behavior with different stopword lists and stemming algorithms. A second priority is removal of the penalty H applied to long documents (Equation 5). This penalty has been effective, but is not justified theoretically. It suggests that the current treatment of tf needs improvement.

PhraseFinder is promising, but much remains to be learned. It is not clear how large a sample is necessary, whether to filter out frequent and/or infrequent associations, or how best to incorporate concepts for query expansion. Building a PhraseFinder database also requires more CPU cycles and disk space than is desirable. (Accessing a PhraseFinder database requires about the same amount of resources as document retrieval.)

Finally, our disappointing results with paragraph retrieval suggest that evidence from paragraphs is only marginally useful in document retrieval. However, we have recently experienced more success with overlapping fixed-length passages of 200-300 words [2]. The apparent explanations are that heuristics for identifying paragraphs are imperfect, and that authors are not consistent in their use of paragraphs.

Acknowledgements

We thank Bob Krovetz, David Haines, Stephen Harding, Yufeng Jing, Michelle LaMar, Dan Nachbar and Margie Connell for their assistance in the work described here. This research was partially supported by the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst.

References

[1] W. R. Caid, S. T. Dumais, and S. I. Gallant. Learned vector-space models for document retrieval. *Information Processing and Management*, (this issue).

[2] J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, Dublin, Ireland, 1994. Association for Computing Machinery.

[3] J. P. Callan and W. B. Croft. An evaluation of query processing strategies using the TIPSTER collection. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–356, Pittsburgh, PA, June 1993. Association for Computing Machinery.

[4] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[5] Kenneth Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, pages 136–143, 1988.

[6] W. Bruce Croft and Howard R. Turtle. Text retrieval and inference. In P. Jacobs, editor, *Text-Based Intelligent Systems*, pages 127–156. Lawrence Erlbaum, 1992.

[7] David Haines and W. B. Croft. Relevance feedback and inference networks. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–11, Pittsburgh, PA, June 1993. Association for Computing Machinery.

[8] D. Harman, editor. *The Second Text REtrieval Conference (TREC2).* National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD, 1994.

[9] Y. Jing and W. B. Croft. An association thesaurus for information retrieval. In *RIAO 4 Conference Proceedings*, New York, October 1994.

[10] B. Sundheim, editor. *Proceedings of the Third Message Understanding Evaluation and Conference.* Morgan Kaufmann, Los Altos, CA, 1991.

[11] H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[12] H. R. Turtle and W. B. Croft. A comparison of text retrieval models. *Computer Journal*, 1992.

[13] Howard R. Turtle and W. Bruce Croft. Efficient probabilistic inference for text retrieval. In *RIAO 3 Conference Proceedings*, pages 644–661, Barcelona, Spain, April 1991.