An Overview of the INQUERY System as Used for the TIPSTER Project

J. Broglio, J.P. Callan
and W. Bruce Croft

CMPSCI Technical Report 93-85
November, 1993

An Overview of the INQUERY System as Used for the TIPSTER Project

John Broglio *James P. Callan* *W. Bruce Croft*

Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610, USA
{broglio, callan, croft}@cs.umass.edu

1. Description of Final System

1.1. Approach

The TIPSTER project in the Information Retrieval Laboratory of the Computer Science Department, University of Massachusetts, Amherst (which includes MCC as a subcontractor), has focused on the following goals:

- Improving the effectiveness of information retrieval techniques for large, full-text databases,
- Improving the effectiveness of routing techniques appropriate for long-term information needs, and
- Demonstrating the effectiveness of these retrieval and routing techniques for Japanese full text databases [5].

Our general approach to achieving these goals has been to use improved representations of text and information needs in the framework of a new model of retrieval. This model uses Bayesian networks to describe how text and queries should be used to identify relevant documents [7, 4, 8]. Retrieval (and routing) is viewed as a probabilistic inference process which compares 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 [1]) 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 feature 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 and 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 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. These operators implement different methods of combining evidence.

Some of the specific research issues we are addressing are morphological analysis in English and Japanese, word sense disambiguation in English, the use of phrases and other syntactic structure in English and Japanese, the use of feature recognizers (for example, company, country and people 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 a lot 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.

1.2. Processing Flow

The main processes in INQUERY are document indexing, query processing, query evaluation and relevance

feedback. We will give a brief description of these processes.

In the document indexing process, documents are parsed and index terms representing the content of documents are identified. INQUERY supports a variety of indexing techniques including simple word-based indexing, indexing based on part-of-speech tagging and phrase identification, and indexing by domain-dependent features such as company names, dates, locations, etc. The last type of indexing is a first step towards integrating detection and extraction systems.

In more detail, the document structure is used to identify which parts will be used for indexing. The first step of this process is to scan for word tokens. Most types of words (including numbers) are indexed, although a stopword list is used to remove very common words. 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"). Words are then stemmed to conflate variants. We have developed a new stemming algorithm that has a number of advantages for operational systems. A number of feature recognisers written with the UNIX utility `flex` are then used to identify objects such as company names and mark their presence in the document using "meta" index terms. A company name such as IBM in the text, for example, will result in a meta-term #COMPANY being recorded at that position in the text. The use of these meta-terms extends the range of queries that can be specified. This completes the usual processing for document text.

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, overhead rates are typically about 40% of the original database size.

Query processing involves a series of steps to identify the important concepts and structure describing a user's information need. INQUERY is unique in that it can represent and use complex structured descriptions in a probabilistic framework. Many of the steps in query processing are the same as those done in document indexing. In addition, a part-of-speech tagger is to used to identify candidate search phrases. Domain-dependent features are recognised and meta-terms inserted into the query representation. The relative importance of query concepts is also estimated, and relationships between concepts are suggested based on simple grammar rules. An evaluation of some of the query processing techniques is presented in [2].

INQUERY is also capable of expanding the query using relationships between concepts found by either using manually specified domain knowledge in the form of a simple thesaurus or by corpus analysis. The WORDFINDER system is a version of INQUERY that retrieves concepts that are related to the query. WORDFINDER is constructed by identifying noun groups in the text and representing them by the words that are closely associated with them (i.e. occur in the same text windows). Concept "documents" are then stored in INQUERY. This technique has shown considerable promise in retrieval experiments.

The query evaluation process uses the inverted files and the query represented as an inference net to produce a document ranking. The evaluation involves probabilistic inference based on the operators defined in the INQUERY language. These operators define new concepts and how to calculate the belief in those concepts using linguistic and statistical evidence. We are constantly experimenting with and refining these operators (for example, the operator defining a phrase-based concept) in order to improve retrieval performance. The efficiency of retrieval is comparable to commercial information retrieval systems.

The relevance feedback process uses information from user evaluations of retrieved documents to modify the original query in ad-hoc retrieval or routing environments. The INQUERY system, because it can represent structured queries, supports a wide range of learning techniques for query modification [6]. In general, new words and phrases are identified in the sample of relevant documents. These are added to the original query and all the terms in the query are then reweighted. With the amount of relevance information available in TIPSTER, relatively simple automatic techniques appear to produce good levels of effectiveness. We are also investigating the effect of using more limited information and more complex learning techniques, such as neural networks.

The Japanese version of INQUERY only differs from the English version in the low-level language processing and some aspects of the interface. We have carried out experiments using both character and word-based representations of documents in combination with word-based processing of queries that are represented using the INQUERY language. These experiments have shown that character-based representations, which are efficient to produce, are surprisingly effective.

1.3. Description of Key Subsystems

As described above, INQUERY has five key subsystems:

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 1: Stemming example: Query text.

- document indexing,
- WORDFINDER,
- query processing,
- query evaluation,
- relevance feedback, and routing.

Each subsystem is described in more detail below.

Document Indexing: The text-processing modules for document indexing are frequently described as *document parsers* or *document parsers and feature recognizers*. The purpose of all document parsing modules is to generate transactions whereby words, or terms, and their locations are stored in the document indexes. The phases of document parsing and term recognition are (1) layout analysis, (2) lexical analysis, (3) syntactic analysis, and (4) concept identification. Each of these phases may generate multiple transactions for each document, each transaction recording *term*, *document* and the *locations* in the document where term is found.

INQUERY has a set of default text processing modules. We will discuss the default module behavior, but these modules can be easily replaced if some behavior other than the default is desired.

The minimal document *layout analysis* must identify the beginnings and end of documents, and the beginning and end of each text segment. Text is distinguished from formatting information, cataloguing information or any information to be excluded for retrieval purposes. Optionally, title, document identification code and other special fields may be identified, and a document layout format may be enforced. The default parser relies on a subset of Standardized General Mark-up Language (SGML), but any layout analysis module may be substituted in order to recognize whatever document style is required. The only requirements are that the layout analysis module indicate to the indexing system when to begin and end indexing for that document, and which section(s) to treat as text to be indexed.

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 2: Stemming example: Query text, after stopword and stop-phrase removal, as stemmed by the Porter stemmer.

marketing strategy carry #usa company #company agriculture chemical report prediction market share chemical report market statistic marketing agrochemic #usa pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

Figure 3: Stemming example: Query text, after stopword and stop-phrase removal, as stemmed by the KSTEM stemmer.

The function of a *lexical analysis* module is to identify and record word boundaries, recognize *stopwords* and *stem* the words as desired, and generate transactions so the words will be indexed 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 or content extraction). These *stopwords* are not indexed, although they are retained in the text so that subsequent textual analysis (syntactic analysis, feature recognition) may make use of them. Stemming is performed to conflate words that have the same root form or stem, in spite of different endings (Figures 1 and 2). The current version of INQUERY uses the Porter stemmer.

A new stemmer, which is more sensitive to the true morphological word stems, has been developed (Figure 3). The KSTEM stemmer has certain advantages over the Porter stemmer. The Porter stemmer produces strings that are not necessarily English words. Because the stemmed representation is sometimes unintelligible, it is difficult to get feedback from a human user by presenting a processed query for modification. In addition, the Porter stemmer sometimes conflates words that are related by only spelling. Thus the phrases "a conflict over foreign policy" and "a conflict between foreigners and the police" would produce the same query network with the Porter stemmer.

The KSTEM stemmer always uses English words as its

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 4: Indexing example: Original document text.

stemming output. It is more conservative as well, conflating only words that are on the same derivational path.

Concept recognition is an important step in automatic classification and glossary construction. The base set of recognizers which are delivered with INQUERY include:

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

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

Company name recognizer: For each citation of a company in the text, generates a transaction for the special term #COMPANY.

Person name recognizer: For each mention of an identifiable person's name in the text, generates a transaction in a special person file.

Figures 4 and 5 illustrate the role that these recognizers play in document indexing.

WORDFINDER: WORDFINDER is a process, analagous to document indexing, in which simple noun groups are indexed by the words that occur near them in the document text. The result of this indexing process is a database in which one or more words can be used to retrieve simple noun groups. This database can be used for query expansion, as a method of adding *concepts* that are related to a set of query terms.

WORDFINDER is based on the assumption that concepts that have a similar lexical context may be related semantically. For example, the words "connectionism" and "neural networks" occur in similar lexical contexts, but rarely in the same documents. The semantic relationship captured is not necessarily synonymy, because WORDFINDER also might relate "connectionism" and "back propagation", which co-occur but have different meanings.

The major steps in constructing a WORDFINDER database are:

John Davenport #PERSON, 52 years old, appointed chief executive officer international telecommunications concern U.S. #USA subsidiary, Cable Wireless North America Inc. #COMPANY Mr. Davenport, #PERSON succeeds John Zrno, #PERSON currently general manager group's operations Bermuda #FOREIGNCOUNTRY.

Figure 5: Indexing example: Document text indexed.

1. Extract simple sequences of nouns, using part of speech tagging (e.g. [3]).
2. For each noun sequence, collect the significant words that occur in any document within a window of n words containing the sequence. The words collected form a feature vector representing the lexical context of the sequence.
3. Queries are expanded by adding noun groups that are close in feature space to the words in the original query (see Figures 6 and 7).

In Figures 6 and 7 the floating point numbers to the left of each query expansion term are *belief* values. The absolute magnitude of a belief value is not meaningful.

Query:115.1 : Impact of the 1986 Immigration Law - will report specific consequence consequences of the U.S.'s Immigration Reform and Control Act of 1986.

0.511462	illegal immigration
0.501936	illegals
0.499120	undocumented aliens
0.498964	amnesty program
0.498054	immigration reform law
0.492453	editorial-page article
0.490993	naturalization service
0.489448	civil fines
0.488754	new immigration law
0.487762	legal immigration
0.487187	employer sanctions
0.483245	simpson-mazzoli immigration reform
0.482687	statutes
0.480449	applicability
0.480222	seeking amnesty
0.478625	legal status
0.478437	immigration act
0.477798	undocumented workers
0.475995	guest worker
0.475995	sweeping immigration law

Figure 6: Query expansion example: Concepts discovered automatically for topic 115.

Query:132.1 : "Stealth" Aircraft - will provide cost, technical, and/or performance data on U.S. "stealth" aircraft projects.

0.529560	northrop corp.
0.528570	tactical fighter
0.525970	aerospace companies
0.520621	flying wing design
0.519889	enemy radar
0.517714	stealth bomber
0.517441	development program
0.515699	radar-evading aircraft
0.514796	bat-winged aircraft
0.513967	cost overruns
0.512299	expensive plane
0.508889	stealth fighter
0.506980	radar-evasion standards
0.505602	full-scale production
0.505052	palmdale
0.503255	radar-evading
0.503105	pentagon official
0.500766	flying wing
0.498029	air force officials
0.496971	development costs

Figure 7: Query expansion example: Concepts discovered automatically for topic 132.

It is the result of the complex process of query evaluation combining term weights, document and collection frequencies and query network structure. Within in the same query expansion set, however, the relative magnitude of belief values is significant. Further research will determine if a weighting scheme for expansion terms can be developed based on the belief values.

There is an advantage to using WORDFINDER for automatic query expansion. Since the query expansion concepts are derived from the text of the collection itself, WORDFINDER automatically creates collection-specific concept links. Further research will compare the value of generic WORDFINDER databases to collection-specific ones.

WORDFINDER is sensitive to several parameters, including the size of the window, decisions about what words to include in the window, and whether low and/or high frequency noun groups are removed from the database. Experiments have shown that getting these parameters right is important to the effectiveness of the concepts returned. The current implementation of WORDFINDER shows promise on the TIPSTER data, but further work is necessary to build a WORDFINDER that would be effective on a variety of document collections.

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 interpreted by text processing modules which ensure that the handling of lexical analysis and concept recognition match that of the indexing subsystem. The resulting modified queries are converted to the structured query language by applying the #SUM operator to the terms in the query. Equations 1–6 (below, in the discussion of the Retrieval engine) describe #SUM and the other operators in INQUERY's structured query language. Query operators permit the text-processing system or the user interface to supply structural information with the query, including phrase grouping or proximity requirements. Query text is generally converted to lower case and checked for stopwords and stemmed, if necessary, to produce canonical word forms.

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. All text processing is experimental and the sequence of operations is adjusted frequently as more is learned about the effects of this processing. Text processing methods which prove successful with queries can then be integrated into the indexing process. This reduces the need to repeatedly index large document collections in order to make small experimental adjustments.

For this reason, much TIPSTER query processing is performed as preprocessing rather than in the body of the INQUERY program. Preprocessing stages are made up of `sed`, `awk`, `flex` (or `lex`) scripts and C code. Query text processing that has been integrated into INQUERY is made up of `flex` feature analyzers.

Currently INQUERY has a small number of internal query text processors. One handles hyphenated words and groups of capitalized words by enclosing them in these proximity operators. Orthographic clues such as hyphenation and capitalization, when reliable, are very good clues to phrasal grouping. Hyphens are generally discarded during indexing so that an expression such as *Iran-Contra* or *voice-activated* become *Iran Contra* and *voice activated*, respectively. In query processing, the symmetrical procedure is to remove the hyphen and to phrase the words as `#1( Iran Contra )` or `#1( voice activated )`, requiring that the two words be found adjacent in a document for the query operator to be satisfied. Groups of capitalized words are similarly phrased so that the phrase *House of Representatives* in a query is transformed to `#3( House Representatives )`.

<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.
<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 8: Query processing example: Original query.

Another of INQUERY's query text processors removes phrases that discuss the retrieval process rather than the desired material, such as "customer reports that..." or "A relevant document must contain...". This behavior is domain dependent, so the program's rules should be modified or augmented for different types of use.

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 (e.g.,

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

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

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

The input and output description for a processor or preprocessor is the same: query text. There are no restrictions. However, text which is tagged with part of speech tags contains more information and more "noise". So, for each query text preprocessor, there are two versions, one which operates on tagged text and one for untagged text.

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 (see Figure 8) 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. Currently we use the tagger developed by Ken Church [3]. This tagger is proprietary software and not available with INQUERY. We have developed our own POS tagger, which we are beginning to use now. 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 replacement is to remove phrases which resemble noun phrases syntactically but which are really syntactic operators (e.g., phrasal prepositions) with no substantive content.

When the text is tagged and the potentially irrelevant material has been removed, syntactically-based noun group capture is performed. Certain kinds of noun phrase patterns are enfolded in a #PHRASE operator (Figure 9):

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

```
#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) 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
2.0 #NOT(#FOREIGNCOUNTRY) )
```

Figure 9: Query processing example: Automatically processed query.

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 the words *company*, *not U. S.* or a restriction in the nationality section of the <fac> field to U.S. or other nationality. 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 9).

There is some recognition of simple time expressions, such as *since 1984* which are expanded to the set of years which 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. In addition, 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 9: 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 9),

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 the word *South Africa* appears as a key concept (and *company* appears somewhere in the query), then the preprocessor would produce the term `#UW50( #COMPANY #1( south africa))` which would match any document

```
#WSUM (1.0
2.0 #UW50 (Control of Insider Trading )
3.0 #3 ( Insider Trading )
1.0 #3( securities law) 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 #uw50( #syn(bill law regulation rules) insider trading)
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)
2.0 #NOT ( #FOREIGNCOUNTRY) )
```

Figure 10: Query processing example: Manually modified query.

which had a company name within fifty words of *South Africa*.

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 10). This has proved to be generally effective in increasing the quality of retrieval results.

Retrieval Engine: Once a query net is formed, the retrieval subsystem can rank documents according to the belief that they are relevant to the query. INQUERY's methods of combining evidence are summarized below, and are documented more fully in [9].

$$bel_{not}(Q) = 1 - p_1 \quad (1)$$

$$bel_{or}(Q) = 1 - (1 - p_1) \cdot \ldots \cdot (1 - p_n) \quad (2)$$

$$bel_{and}(Q) = p_1 \cdot p_2 \cdot \ldots \cdot p_n \quad (3)$$

$$bel_{max}(Q) = \max(p_1, p_2, \ldots, p_n) \quad (4)$$

$$bel_{wsum}(Q) = \frac{(w_1 p_1 + w_2 p_2 + \ldots + w_n p_n) w_q}{(w_1 + w_2 + \ldots + w_n)} \quad (5)$$

$$bel_{sum}(Q) = \frac{(p_1 + p_2 + \ldots + p_n)}{n} \quad (6)$$

Node belief scores are calculated as a combination of term frequency (tf) and inverse document frequency (idf) weights. The values are normalized to remain between 0 and 1, and are further modified by tf and belief default values which the user may define at program invocation. Calculation of a belief for a given query operator is dependent on the type of operator and the number and belief in its arguments as presented in Equations 1-6.

Relevance Feedback and Routing: 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 [6]. Experiments have been conducted with a variety of algorithms for term selection and weighting.

Early experiments [6] 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 at 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).

Our approach to the *routing* portion of our TIPSTER 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 in the Query Processing section above. 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 the 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

#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 11: Routing profile created automatically from relevant documents.

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 set of pairs are ranked by the formula

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

$|R|$ is the number of 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 proximity operators with $n = 5$, and 20 proximity operators with $n = 50$, were added to the 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. Unofficial 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 11 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 to reach a given level of performance. Using relevant documents retrieved by many systems (e.g. the TREC 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 relevant documents are available for analysis, a combination of automatic query formation and manual query construction by a user is superior to either approach alone.

1.4. Hardware/Software Requirements

INQUERY was developed to run under the UNIX operating system, on workstations manufactured by Digital

Equipment Corporation (DEC), and SUN Microsystems (SUN). It has also been ported to the MS-DOS operating system (with and without the Windows graphical user interface) on personal computers containing the Intel 386 and 486 microprocessors. These hardware platforms include 16, 32 and 64 bit architectures. Ports to other computers and operating systems are not expected to present any serious problems.

The INQUERY system consists of several major subsystems, as described in Section above. The application programmer's interface (API) for document retrieval applies to all INQUERY document collections, whether local or remote (available on another machine on a network). An application can be configured for a client/server environment with just a few compile-time and link-time decisions. Clients can run on any of the platforms mentioned above. The server processes are constrained currently to run on a UNIX workstation. The DecNet and TCP/IP networking protocols are both supported. Additional network protocols can be added easily.

Compile-time and link-time options are also available to enable a programmer to use only those modules necessary for a particular application. For example, if a client program will access only remote databases, the code for accessing local databases can be eliminated, resulting in a substantially smaller executable program.

The amount of memory and disk space required for document retrieval depends on the size of the document collection. For a collection of N bytes, INQUERY requires about $5N$ bytes of disk space to build its document database. Once the database is built, INQUERY requires about $1.5N$ bytes of disk space to store the document database. Memory requirements are more difficult to predict, because they depend upon the characteristics of the document collection, the complexity of the queries, and the hardware characteristics of the computer. For UNIX workstations, a very rough estimate is that INQUERY requires about $\frac{N}{15}$ bytes of virtual memory. A reasonable amount of physical memory is $\frac{N}{60}$. Therefore a 2 gigabyte collection would need about 135 MB of virtual memory and about 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 can be reduced. Experiments have been conducted with an in-memory approach to document indexing that eliminates the need for a separate sort of the indexing transactions. 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 new 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.

INQUERY is implemented in the ANSI standard version of the C programming language. The `flex` and yacc programs are required to create the lexical scanners and parsers needed during document indexing and query processing. These scanners and parsers are provided with each release of INQUERY, so `flex` and yacc are not needed except to customize the system.

1.5. Speed & Throughput

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.

1.6. Key Innovations of Final System

The following are the key innovations developed during the course of the TIPSTER project:

1. An inference net retrieval model for large, heterogeneous databases.
2. Query processing techniques that transform complex queries into INQUERY structures.
3. Query expansion techniques using WORDFINDER.
4. New techniques for handling noun phrases in the retrieval model.
5. Techniques for automatic construction of profiles for routing.
6. Techniques for combining document and paragraph-level representations.
7. Techniques for combining queries.
8. A new stemming algorithm (KSTEM).

9. Techniques for integrating feature extraction and indexing.

10. New approaches to word indexing.

11. Indexing and query processing techniques for Japanese.

2. Original Project/System Goals

The original goal of this project as described in the University of Massachusetts proposal was to design and implement algorithms for document detection that would achieve significant improvements in retrieval effectiveness relative to conventional techniques. We defined significant improvements as at least 10% increases in average precision figures. In addition, these algorithms should be portable, extensible, trainable, improvable, and robust.

To achieve these goals, we decided to focus on algorithms for text representation, acquisition of information needs, and retrieval, starting from a basis of statistical retrieval techniques. In particular, we emphasized the development of new text representation techniques based on natural language processing, *coupled* with the development of a probabilistic retrieval model using Bayesian inference networks.

We proposed to do work in the following areas:

1. Text Representation.
   (a) Theories of text representation quality.
   (b) Morphological processing - improving the indexing technology for English and providing basic indexing for Japanese.
   (c) Word senses - reducing ambiguity in text representation by identification of word senses in queries and documents.
   (d) Syntactic phrases - indexing by phrases formed using syntactic criteria, augmented using cluster analysis.
   (e) Statistical phrases - indexing by groups of index terms formed using statistical criteria, both in English and Japanese.
2. Retrieval Models.
   (a) Theories of retrieval viewed as inference.
   (b) Basic inference networks - implementing and evaluating retrieval and routing strategies in English and Japanese.
   (c) Combining multiple sources of evidence - using inference networks to combine representations produced by the text representation research.
3. Acquisition of Information Needs.
   (a) Query formulation - acquiring additional information to improve the accuracy of queries in English and Japanese.
   (b) Relevance feedback - developing feedback algorithms for inference networks, for both English and Japanese.
   (c) Profile formation - investigating strategies for effective long-term profile formation using relevance feedback.

3. Evolution of System Over 2 Years

Over the two years of the project, the project goals and the main areas of work (text representation, retrieval models, and acquisition of information needs) have not changed. The focus of some of the work in those areas has shifted somewhat.

In the text representation area, considerable work has been done on morphological processing for both English and Japanese. In English, this work has been directed at improving the basic indexing and stemming algorithms. In Japanese, character-based indexing has emerged as an alternative to word-based indexing. There has also been substantial work done on phrases. The shift here has been from parsing documents to locate phrases at index time, to parsing queries to determine which phrases are important and then looking for evidence of the presence of those phrases in documents. Two major changes have been the lowering in priority of word sense disambiguation algorithms, and a different approach to clustering. Our initial experiments with word sense disambiguation were not promising and for that reason we have delayed the implementation of the algorithm until the end of the project. With regard to clustering, we are now grouping nouns and noun phrases by the similarity of their contexts in the test collections. The system for concept retrieval and comparison that results from this approach is WORDFINDER.

The retrieval model area has been essentially carried out as planned. Experiments evaluating retrieval and routing have been done on schedule, and the idea of combining multiple sources of evidence has turned out to be central both in our work, and in other aspects of the TIPSTER project. One aspect of this that we have paid particular attention to is paragraph-based retrieval.

In the area of information need acquisition, we have carried out extensive experiments with relevance feedback

and routing as planned. In query formulation, we have placed the most emphasis on experiments with simple manual modifications to automatically processed queries and query expansion.

4. Accomplishments

The TIPSTER 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.

Indexing:

- Stopwords are sometimes necessary (e.g. *Ms. The, sit-in*). In order to reduce the number of queries that fail due to incomplete indexing, we have extended the basic indexing algorithms to include indexing of stopwords when they are capitalized but not at the start of sentences, and when they are joined to other words.
- It is sometimes difficult to decide what is a word (e.g. numbers, special characters). We have carried out a number of experiments to determine the most flexible and efficient way of indexing word tokens.
- "Real" paragraph boundaries often do not indicate content shift in documents. Our experiments with paragraph-based retrieval indicate that real paragraphs are no more effective than text windows, although there are collection-specific exceptions, such as detecting Wall Street Journal articles that cover multiple short topics.
- Feature recognition can add significant overhead. We have done the first experiments on the impact of including simple extraction (e.g. company names, dates, locations) in the indexing process. We have attempted to reduce the indexing overhead to a minimum in order to support high-volume updates (e.g. routing).
- "Justified" stemming is hard (effective stems vs. understandable stems). We have developed a new stemming algorithm that produces much more understandable stems than the current standard (Porter). The effectiveness of the new algorithm varies across collections and we are continuing to work towards consistent improvements.

Query processing:

- Sophisticated query processing produces significant improvements. We have developed a variety of query processing techniques that together have improved the overall system effectiveness considerably.
- Large queries resulted in less emphasis on word disambiguation. The TIPSTER topics are much longer than typical IR queries and the mutual disambiguation produced by the presence of so many terms has made word sense disambiguation a marginal technique. We continue to study this technique with other collections.
- Large queries also makes query expansion difficult. Simple query expansion techniques, such as using a general thesaurus, are not effective in this environment.
- Automatic query expansion still looks promising. Query expansion based on the WORDFINDER system has produced the most significant results of their type to date.
- Extracting query terms/phrases from the narrative is hard. The abstract nature of the narrative section of the TIPSTER topics makes automatic processing difficult. This is a promising area for future research.
- Feature extraction/recognition is 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.
- Manual queries can improve results, but usually only in combination with automatically processed queries. In general, we have shown that automatically processed queries are competitive with handcrafted queries.

Retrieval:

- Improvements depend heavily on baseline. In general, it is easy to get large percentage improvements if a baseline search with poor performance is used. The improvements obtained using INQUERY are often small, but the overall effectiveness is consistently better than other approaches.
- Phrases were not as effective as on other collections. Despite considerable efforts on developing the phrase model, the overall improvements obtained

using phrases are small (but consistent). We are continuing to work on this issue.

- Estimation was a significant problem for large, heterogeneous databases. The estimation functions used for previous, smaller test collections proved to be inadequate for TIPSTER. We have developed new forms of these functions which have resulted in significant improvements.
- Paragraph-level retrieval can help in combination with document-level retrieval. In full-text collections, the notion of a local match and a global match is important. We have shown that paragraph-level matching can produce significant improvements in effectiveness in two situations. One is when paragraph-level connections between query concepts are specified manually, the other is when automatic paragraph-level matching is combined (in the INQUERY framework) with document-level matching.

Routing:

- Automatic construction of routing profiles has consistently outperformed manually specified profiles in our experiments. Combining these forms of profile results in further improvements.
- Proximity pairs are important in the automatic profile. We compared simple word-based learning with learning structure in the form of phrase and paragraph-level proximities and found that the structured profiles perform better.
- Routing is different than relevance feedback. Techniques that were superior in relevance feedback experiments (small amounts of training data), have not been the best in routing experiments (large amounts of training data).
- Amount of training data has significant, but limited effect on performance. We have shown that good performance can be obtained with limited amounts of training data or relevance judgements. This has important implications for practical applications.

Japanese:

- INQUERY works well with Japanese with minor changes. The only differences between the Japanese and English versions of INQUERY are the modules for the morphological processing and the interface. The operators in the query language are identical, although there is some evidence that a Japanese-specific phrase operator may be more effective.
- Character-based indexing can be competitive with word-based indexing. Indexing all characters is much faster than segmenting Japanese into words, and our retrieval experiments have shown the effectiveness levels to be similar. The best performance was obtained using combinations of both representations.
- Query processing is at least as important in Japanese as in English. Parsing a Japanese query and constructing the appropriate INQUERY query from that parse is a crucial part of getting effective performance, particularly with character-based indexing. Experiments with different approaches to query processing are continuing as more topics are obtained.

5. Evaluation Summary

In this section we discuss the results obtained in the 24 month evaluation of the TIPSTER project.

Table 1 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. 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 were judged to be important, and by deleting terms judged to be not relevant. For INQ041, the query modifier was allowed to add any structure that was thought to be appropriate. Examples of this type of structure are paragraph-level proximity constraints between query terms. The results show that manual modification has little effect on performance, with the main effect being an increase in precision at low recall levels for INQ041. In previous evaluations, manual modifications resulted in significant improvements. The result reported here may be due to the difficulty of the topics in the third set. The results for INQ010 also indicate that using NLP techniques to analyze the Narrative section of a topic may not improve the query. The results labelled INQ015 and INQ016 were for an early version of the WORDFINDER query expansion system. Although these results show no significant differences, other WORDFINDER results are presented later.

Given that automatically processed queries had similar performance to manually produced queries, the next experiment combined these two version of the information need using the INQUERY framework. The result of this combination (INQ044) was significantly better than either of the individual sets of queries, and retained the high precision of the manual queries with the high recall of the automatic queries.

Table 1: Ad-hoc results.

INQ009 – baseline
INQ010 – official manual (simulated NLP)
INQ015 – Wordfinder 1
INQ016 – Wordfinder 2
INQ041 – Unofficial manual

	Precision (% change) – 50 queries								
Recall	INQ009	INQ010		INQ015		INQ016		INQ041	
0	82.7	77.6	(−6.2)	82.5	(−0.3)	82.8	(+0.0)	87.4	(+5.6)
10	60.0	63.0	(+5.1)	60.3	(+0.6)	60.2	(+0.3)	63.1	(+5.3)
20	54.3	56.1	(+3.2)	54.3	(−0.1)	54.2	(−0.2)	57.1	(+5.0)
30	48.4	49.3	(+1.9)	48.5	(+0.3)	48.4	(+0.0)	51.1	(+5.6)
40	44.0	43.8	(−0.4)	44.1	(+0.2)	43.9	(−0.2)	43.3	(−1.6)
50	38.0	38.2	(+0.4)	38.3	(+0.7)	38.2	(+0.4)	36.9	(−3.1)
60	33.0	32.3	(−1.9)	33.1	(+0.4)	33.2	(+0.7)	30.8	(−6.4)
70	27.4	26.6	(−3.0)	27.6	(+0.8)	27.5	(+0.2)	25.0	(−8.7)
80	21.4	21.0	(−1.8)	21.7	(+1.3)	21.6	(+1.0)	19.3	(−9.8)
90	14.5	13.9	(−4.3)	14.6	(+0.9)	14.6	(+0.6)	13.3	(−8.6)
100	3.3	3.3	(+0.3)	3.3	(−0.1)	3.3	(−0.1)	1.6	(−49.7)
avg	38.8	38.6	(−0.5)	38.9	(+0.3)	38.9	(+0.2)	39.0	(+0.5)

Another set of experiments that were done with the ad-hoc queries was to combine the results of document-level representations with paragraph-level representations. This was done using the #WSUM operator in INQUERY. A significant 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, and we are currently working on improving this.

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 new terms (thirty were added) 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 this is small for this experiment due to the relatively poor performance of the manual queries. 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. Both of these produced significant improvements. The best result (INQ024) was from a run where the original query was ignored and all terms came from relevant documents.

Tables 5 and 6 show additional results using WORDFINDER. TipC and TipT were the results of using topics 51-100 expanded using the best 5 concepts from WORDFINDER. These queries were run against the first TIPSTER disk. For TipC, the query used to search WORDFINDER was the concepts from each topic, whereas for TipT, it was the Description field. The results show substantial improvements, and is one of the best results ever obtained for automatic query expansion. The results also show that even the relatively short Description field retrieved good concepts for expansion. These results were all obtained using a training set of 250,000 documents from WSJ, AP and Ziff to build the WORDFINDER database. The results in T3T, T3C and T5C were obtained using a smaller 50,000 document collection as the basis for WORDFINDER. The results, although not as good as with the larger database, are still significant. It appears that a smaller text window gives better performance.

Table 7 gives an indication of the progress of the INQUERY system over the 2 year time frame of the TIPSTER project. The baseline in this case has had all sys-

Table 2: Ad-hoc combination results. INQ044 was a combination of baseline automatic (INQ009) and manual (INQ041). Combination done using #SUM operator.

	Precision (% change) – 50 queries				
Recall	INQ009	INQ041		INQ044	
0	82.7	87.4	(+5.6)	88.0	(+6.4)
10	60.0	63.1	(+5.3)	64.1	(+6.9)
20	54.3	57.1	(+5.0)	58.5	(+7.6)
30	48.4	51.1	(+5.6)	52.8	(+9.2)
40	44.0	43.3	(–1.6)	46.4	(+5.5)
50	38.0	36.9	(–3.1)	40.8	(+7.2)
60	33.0	30.8	(–6.4)	35.7	(+8.3)
70	27.4	25.0	(–8.7)	29.5	(+7.6)
80	21.4	19.3	(–9.8)	23.5	(+10.0)
90	14.5	13.3	(–8.6)	16.0	(+10.2)
100	3.3	1.6	(–49.7)	3.5	(+8.3)
avg	38.8	39.0	(+0.5)	41.7	(+7.5)

tem refinements from this period removed. In general, these improvements are consistently better than 10%, but in the case where WORDFINDER is included, there is nearly a 30% improvement.

Table 8 shows that even small changes can still produce improvements. We have in general resisted tuning INQUERY to the TIPSTER collection. We have, however, had to change the probability estimation formulae considerably to produce good results for a large, heterogeneous database. The improvement in Table 8 is the result of running experiments on the earlier parts of TIPSTER, and more improvements of this kind can be expected.

Tables 9 and 10 give the results of experiments on a small Japanese database using 27 of our own queries and relevance judgements. The results compare word-based and character-based indexing with different forms of query processing. The results show that word-based indexing performs well at low recall levels, but that averaged over all recall levels, it is significantly lower. The best performance has been obtained using a combination of both representations (this result is not shown here).

An overall summary of what has been learned over the two years of the TIPSTER project is given in the Accomplishments section. Here, we will very briefly summarize the major results:

- INQUERY has been shown to be an effective, flexible and efficient retrieval and routing engine.
- TIPSTER topics are very different to typical IR queries, and new query processing techniques produced good results.
- IR techniques, which are primarily based on words and statistics, produced very good results when scaled up for the TIPSTER database and topics.
- For ad-hoc queries, automatic processing produces very good results. Manual modification can result in significant benefits, but topics 101-150 were different in that respect.
- Paragraph-based retrieval can help performance, but only when used in combination with document-based results.
- Automatic query expansion techniques produced encouraging results, but more work is needed to refine the techniques and understand the size of training set needed.
- Automatic relevance feedback produces very good results for routing when there are large numbers of relevance judgements available. Manual queries are not competitive, but can help in combination.
- Combining document representations and queries, for which INQUERY was designed, can result in very significant performance improvements. This was also shown to be true for Japanese.

References

[1] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the*

Table 3: Paragraph-based retrieval

Queryid (Num): (BASELINE)	INQ009 (DOC)	INQ011 (PAR)	INQ012 (DOC+PAR/2)
Total number of documents over all queries			
Retrieved:	50000	50000	50000
Relevant:	11547	11547	11547
Rel_ret:	8651	7814	8851
Interpolated Recall - Precision Averages:			
at 0.00	0.8275	0.7067	0.8400
at 0.10	0.5945	0.4722	0.6101
at 0.20	0.5383	0.4218	0.5512
at 0.30	0.4775	0.3673	0.4985
at 0.40	0.4290	0.3170	0.4468
at 0.50	0.3720	0.2659	0.3864
at 0.60	0.3133	0.2055	0.3265
at 0.70	0.2316	0.1393	0.2482
at 0.80	0.1388	0.0758	0.1652
at 0.90	0.0588	0.0293	0.0758
at 1.00	0.0095	0.0046	0.0082
Average precision (non-interpolated) over all rel docs			
	0.3480	0.2511 (-27.9)	0.3629 (+4.3)

Third International Conference on Database and Expert Systems Applications, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[2] James P. Callan and W. Bruce Croft. An evaluation of query processing strategies using the tipster collection. In *Proceedings of the 16th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 347–356. ACM, June 1993.

[3] 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.

[4] 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.

[5] Hideo Fujii and W. B. Croft. A comparison of indexing techniques for Japanese text retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 237–246, 1993.

[6] David Haines and W. B. Croft. Relevance feedback and inference networks. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–11, 1993.

[7] 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.

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

[9] 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.

Table 4: Routing results.

INQ026 - Automatic baseline
INQ020 - Relevance feedback
INQ021 - Relevance feedback + manual query
INQ022 - Manual query
INQ023 - Relevance feedback w/prox
INQ024 - Relevance feedback w/prox and w/o original query

	Precision (% change) – 50 queries											
Recall	INQ026	INQ020		INQ021		INQ022		INQ023		INQ024	
0	78.7	80.7	(+2.5)	84.7	(+7.6)	81.5	(+3.7)	81.1	(+3.1)	82.5	(+4.9)
10	55.7	62.7	(+12.5)	64.9	(+16.6)	61.0	(+9.5)	66.5	(+19.3)	65.8	(+18.1)
20	46.5	54.5	(+17.2)	54.2	(+16.5)	48.0	(+3.4)	58.1	(+25.0)	57.1	(+23.0)
30	40.5	48.9	(+20.6)	48.7	(+20.2)	40.2	(-0.7)	50.4	(+24.4)	50.4	(+24.4)
40	35.5	44.4	(+25.0)	43.7	(+23.2)	35.6	(+0.4)	45.9	(+29.4)	44.9	(+26.5)
50	31.1	39.1	(+25.6)	38.6	(+24.2)	30.9	(-0.7)	41.3	(+32.8)	40.5	(+30.1)
60	26.9	32.1	(+19.3)	33.4	(+24.2)	26.1	(-3.0)	34.6	(+28.6)	35.9	(+33.6)
70	22.6	27.7	(+22.2)	27.9	(+23.2)	22.2	(-2.0)	30.3	(+33.6)	31.5	(+39.0)
80	18.0	22.2	(+23.5)	22.8	(+26.9)	17.9	(-0.5)	23.9	(+33.1)	25.8	(+43.5)
90	11.2	14.4	(+29.0)	14.9	(+33.7)	10.9	(-2.2)	15.0	(+34.5)	17.5	(+56.7)
100	1.5	2.8	(+84.3)	3.0	(+97.4)	2.1	(+42.2)	2.9	(+91.8)	3.4	+127.2)
avg	33.5	39.0	(+16.6)	39.7	(+18.7)	34.2	(+2.3)	40.9	(+22.2)	41.4	(+23.7)

Table 5: Wordfinder results.

Set-2: Baseline using set 2 queries on first TIPSTER database.
TipC: Queries expanded using concepts to search WORDFINDER, first 5 unique concepts added to query, WORDFINDER built using 250,000 documents from WSJ, AP and Ziff.
TipT: Queries expanded using topic and description.

	Precision (% change) – 50 queries				
Recall	set-2	TipC		TipT	
0	87.5	86.8	(-0.8)	87.9	(+0.5)
10	65.8	70.0	(+6.5)	68.9	(+4.8)
20	57.4	62.6	(+9.1)	61.5	(+7.2)
30	51.4	56.5	(+10.0)	56.0	(+9.1)
40	45.8	52.4	(+14.4)	51.0	(+11.4)
50	39.7	45.9	(+15.4)	46.3	(+16.5)
60	34.5	40.8	(+18.5)	41.0	(+19.1)
70	28.6	35.4	(+23.7)	35.4	(+24.0)
80	22.1	28.2	(+27.8)	29.3	(+32.7)
90	14.2	19.6	(+38.1)	19.3	(+35.7)
100	3.4	5.3	(+57.6)	4.8	(+41.9)
avg	40.9	45.8	(+11.8)	45.6	(+11.4)

Table 6: Wordfinder results.

Set-2: Baseline using set 2 queries on first TIPSTER database
T3T: Queries expanded using topic and description to search WORDFINDER, first 5 unique concepts added to query, WORDFINDER built using 50,000 documents from WSJ, AP and Ziff, text window of 3 sentences.
T3C: Queries expanded using concepts
T5C: Text window of 5 sentences

	Precision (% change) – 50 queries					
Recall	set-2	T3T		T3C	T5C	
0	87.5	84.9	(-3.0)	87.2	(-0.3)	87.6	(+0.1)
10	65.8	67.9	(+3.2)	69.8	(+6.1)	68.4	(+4.0)
20	57.4	59.1	(+3.1)	61.3	(+6.8)	60.4	(+5.4)
30	51.4	55.1	(+7.3)	56.1	(+9.2)	55.2	(+7.4)
40	45.8	50.6	(+10.6)	50.8	(+11.0)	50.3	(+9.8)
50	39.7	45.5	(+14.5)	45.2	(+13.8)	45.0	(+13.1)
60	34.5	40.1	(+16.3)	40.1	(+16.4)	39.6	(+15.0)
70	28.6	34.6	(+21.0)	34.7	(+21.5)	34.4	(+20.2)
80	22.1	28.3	(+28.0)	27.8	(+26.0)	27.8	(+25.6)
90	14.2	19.4	(+36.2)	19.1	(+34.6)	19.1	(+34.3)
100	3.4	4.6	(+36.5)	4.8	(+40.6)	4.7	(+39.2)
avg	40.9	44.6	(+8.8)	45.2	(+10.4)	44.8	(+9.4)

Table 7: Improvements during TIPSTER project. Queries for topics 51–100, run against Volume 1 of the collection.

	Precision (% change) – 50 queries		
Recall	set-2-basic	TipC	
0	81.8	86.8	(+6.1)
10	56.4	70.0	(+24.1)
20	48.1	62.6	(+30.2)
30	43.3	56.5	(+30.5)
40	38.5	52.4	(+36.1)
50	34.3	45.9	(+33.8)
60	29.9	40.8	(+36.5)
70	25.1	35.4	(+41.0)
80	19.5	28.2	(+44.6)
90	12.4	19.6	(+58.1)
100	2.5	5.3	(+112.0)
avg	35.6	45.8	(+28.7)

Table 8: Optimizing estimation formulae.

	Precision (% change) – 50 queries		
Recall	INQ009	INQ009a	
0	82.7	82.8	(+0.1)
10	60.0	60.0	(+0.1)
20	54.3	54.9	(+1.0)
30	48.4	49.1	(+1.5)
40	44.0	44.6	(+1.4)
50	38.0	38.8	(+2.1)
60	33.0	33.9	(+2.7)
70	27.4	28.0	(+1.9)
80	21.4	21.7	(+1.4)
90	14.5	14.9	(+2.5)
100	3.3	3.3	(+1.0)
avg	38.8	39.3	(+1.2)

Table 9: Japanese results: Character-based indexing on a database of 1100 documents.

NLQ(C) Character-based natural language query
Short(C) Characters combined at word level
Long(C) Characters combined at phrase level
Joined(C) Characters combined at complex phrase level

	Precision (% change) – 27 queries							
Recall	NLQ	Short (C)		Long (C)		Joined (C)		
10	65.3	67.8	(+3.7)	71.3	(+9.1)	71.3	(+9.1)	
20	61.6	61.9	(+0.5)	62.1	(+0.8)	66.0	(+7.2)	
30	54.7	52.5	(–4.1)	54.3	(–0.9)	56.7	(+3.6)	
40	49.4	47.3	(–4.3)	47.6	(–3.7)	51.5	(+4.2)	
50	47.1	41.6	(–11.6)	41.8	(–11.3)	47.9	(+1.8)	
60	42.4	38.0	(–10.5)	37.2	(–12.4)	42.9	(+1.0)	
70	38.1	34.0	(–10.7)	33.2	(–12.8)	38.3	(+0.4)	
80	34.8	30.3	(–13.0)	29.8	(–14.5)	34.8	(–0.0)	
90	29.4	23.4	(–20.6)	24.8	(–15.7)	29.4	(+0.0)	
100	15.5	15.0	(–3.2)	14.5	(–6.5)	15.5	(+0.0)	
avg	43.8	41.2	(–6.1)	41.7	(–5.0)	45.4	(+3.6)	

Table 10: Japanese results: Word-based indexing on a database of 1100 documents.

NLQ(W): Words (from JUMAN) in natural language query
Long(W): Words combined at phrase level
Joined(W): Words combined at complex phrase level

	Precision (% change) – 27 queries				
Recall	NLQ(W)	Long (W)		Joined (W)	
10	75.4	75.6	(+0.3)	77.7	(+3.1)
20	67.4	67.5	(+0.1)	68.5	(+1.6)
30	53.2	55.6	(+4.5)	54.7	(+2.9)
40	44.8	44.8	(+0.1)	46.7	(+4.3)
50	39.3	38.1	(-3.1)	40.7	(+3.6)
60	36.5	34.1	(-6.4)	36.9	(+1.3)
70	31.0	30.1	(-2.8)	31.3	(+1.1)
80	28.2	26.7	(-5.5)	28.2	(+0.0)
90	20.9	20.1	(-3.7)	20.9	(+0.0)
100	13.1	12.8	(-1.9)	13.1	(+0.0)
avg	41.0	40.5	(-1.0)	41.9	(+2.2)