A Loosely-Coupled Integration of a Text Retrieval System and an Object-Oriented Database System

W. Bruce Croft, Lisa A. Smith*
Computer Science Department
University of Massachusetts, Amherst, MA 01003
Howard R. Turtle
West Publishing Company, St. Paul, MN 55164

Abstract

Document management systems are needed for many business applications. This type of system would combine the functionality of a database system, (for describing, storing and maintaining documents with complex structure and relationships) with a text retrieval system (for effective retrieval based on full text). The retrieval model for a document management system is complicated by the variety and complexity of the objects that are represented. In this paper, we describe an approach to complex object retrieval using a probabilistic inference net model, and an implementation of this approach using a loose coupling of an object-oriented database system (IRIS) and a text retrieval system based on inference nets (INQUERY). The resulting system is used to store long, structured documents and can retrieve document components (sections, figures, etc.) based on their text contents or the contents of related components. The lessons learnt from the implementation are discussed.

1 Introduction

Database systems provide robust, long-term storage of objects and maintain the validity of the data through recovery and concurrency control mechanisms. Current database systems, such as those based on the relational model, can represent objects with complex structure only with difficulty, and often restrict the type of data that can be stored in an object. In response to these shortcomings, object-oriented database systems have been designed specifically to represent complex objects and accommodate user-defined extensions such as new data types [22]. The content-based retrieval capability of these systems, however, is typically limited to selection from a database of objects using Boolean combinations of simple predicates (for example, [14, 2]), although it is in general possible to define additional predicates for specific data types.

In order to provide the powerful document management facilities that are needed in many business applications, it will be necessary to combine the ability of advanced database systems to model, store and maintain documents with complex structure, with more effective retrieval strategies based on information retrieval (IR) models. It may be necessary, however, to extend these models to deal with retrieval situations that are not encountered in databases of simple, abstract-length texts. In an object-oriented database, for example, text objects can be composed of other text objects, and other types of objects, such as tables and figures, may contain little or no useful text. The most effective retrieval strategies for such objects are not known.

The inference net model appears to be a good candidate for a retrieval model for complex objects. This model emphasises the role of probabilistic inference, where multiple sources of evidence are used to assess the likelihood that an object satisfies the user's information need [20]. In the context of complex object retrieval, the inference net model can describe how the meanings of objects are related. The inference net model also allows for a great deal of flexibility in formulating a query and relating the query concepts to the concepts used to describe objects [6].

In a recent paper [8], we described an approach to using the inference net model in an object-oriented environment, particularly for the retrieval of composite

*On leave from Digital Equipment Corporation

15th Ann Int'l SIGIR '92/Denmark-6/92
© 1992 ACM 0-89791-524-0/92/0006/0223...$1.50

and multimedia objects. In this paper, we will describe an alternative model, and a prototype implementation of the model using a loosely-coupled integration of an object-oriented database system (IRIS [21]) and a text retrieval system based on the inference net model (INQUERY [4]).

The goals of the implementation were to demonstrate the capabilities of the retrieval model, to provide a platform for future experiments, and to discover the types of implementation problems that occur with this type of integration. After describing the retrieval model and the architecture of the integrated system, we present some retrieval examples using the system and then discuss limitations of the implementation. The test database that was used consists of dissertations in LaTeX format. We considered this type of document to be representative of long, structured documents described using a markup language.

2 Approaches to Integration

2.1 Database Systems and IR

A number of proposals have been made for incorporating text retrieval functionality in database systems. Blair and others [3, 13, 18] have discussed the implementation of text-oriented applications using standard relational database systems and SQL. These approaches typically have limited representation of complex document structure and restrict the retrieval model to exact matching of Boolean combinations of words.

Lynch and Stonebraker[12] use abstract data types to incorporate text retrieval in the POSTGRES extended relational database system. Although their work is also restricted to exact match searching, the POSTGRES framework could be used to implement more sophisticated retrieval models.

In the database community, related research has been done in the general area of uncertainty in databases. Examples of this type of work are systems that deal with extensions to the relational model to handle uncertain data [11] and uncertain queries [15]. The inference net model described in this paper can address both of these types of uncertainty through the various probabilities represented.

Motro [15] discusses an extension to a relational database system for inexact queries. His VAGUE system incorporates a "similar-to" comparison operator in queries. The retrieval model is based on the vector space model used in IR [17]. Garcia-Molina and Porter [11] discuss a probabilistic relational data model where probabilities are associated with the values of attributes.

Two other related efforts include our work on the interactive retrieval of complex objects [7], and Fuhr's paper on a probabilistic database incorporating relevance feedback [10].

2.2 The Inference Net Approach

In this section, we describe how the inference net retrieval model can be used for retrieval of complex objects. This complex object retrieval model is the basis for the loosely-coupled architecture described in the rest of the paper. By describing this retrieval model, however, we are not solving the whole problem of complex object retrieval. There remains many other issues, such as integration with a complete query language, integration with "structural" search (e.g. [5]), query optimization, and interface design.

The inference net model is described in detail in [19, 20]. It is a probabilistic retrieval model that computes P(I|Object), which is the probability that a user's information need is satisfied given a particular object. Objects are usually considered to contain text, although in the context of complex object retrieval, this is often not the case. We consider an information need as a complex proposition about the content of an object, with possible values true and false. Queries are regarded as representations of the information need. The major difference between the inference net model and other probabilistic models is that it emphasizes the use of multiple sources of evidence to calculate P(I|Object).

To apply the inference net model to the retrieval of complex objects in object-oriented databases, we must specify how the meanings of an object and its subobjects are related. The first step in this process is to define our complex object terminology more precisely. In an object-oriented database, an object is an instance of an object class. The definition for that class of objects will be part of the *class hierarchy* for that system and application. That is, a class definition will inherit *instance variables* and *operations* from its superclass(es) in the class hierarchy. The instance variables of an object may contain references to other objects (called subobjects). These objects may be instances of a number of classes, and may in turn contain references to other objects. It is this variability in structure and content that makes such objects *complex* relative to the simple text abstracts stored in many bibliographic databases.

Figure 1 shows an example of a complex object, which is an instance of a Dissertation object type. This document has a complex structure made up of subobjects of types Chapter, Section, Subsection, Subsubsection, Paragraph, Figure, and Table.

As part of specifying a query for complex object retrieval, users should be able to indicate the required object class. That is, any object class whose instances can be part of a complex object is a valid result. In the case of dissertations, for example, users could specify whether they wanted to retrieve chapters, sections, paragraphs, figures or tables. Objects that are in-

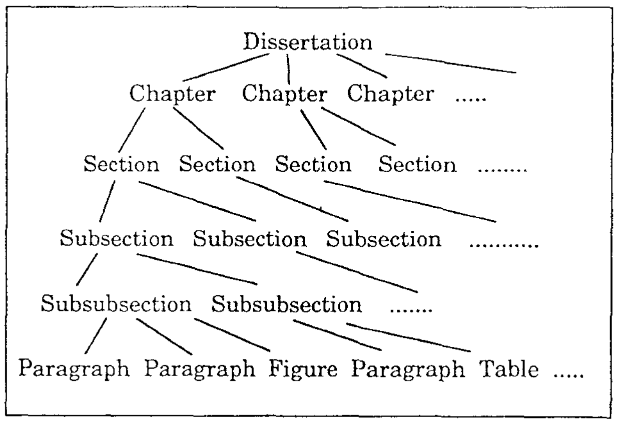

Dissertation
Chapter Chapter Chapter .....
Section Section Section Section ....
Subsection Subsection Subsection ....
Subsubsection Subsubsection ....
Paragraph Paragraph Figure Paragraph Table .....

Figure 1: A complex document type *Dissertation*

stances of the specified class should also be able to be retrieved based on their own content (that is, the values of the instance variables that refer to *primitive objects*, such as numbers, strings and text). Alternatively, their retrieval could be based on the content of their subobjects. These observations suggest possible retrieval models for complex objects based on an inference net.

Figure 2 shows one such model. The network consists of object nodes (o_i's), concept representation nodes (r_k's) and a query node (q). Object nodes in this network correspond to objects with at least some text content (for example, a paragraph or a figure with a caption). These objects can be of many different types and information about object type is not part of this network. It is important to note that the inference net is a separate representation from the object class hierarchy and the complex object structure.

We represent the assignment of a specific representation concept to an object by a directed arc to the representation node from each node representing an object to which the concept has been assigned. A representation node contains a specification of the conditional probability associated with the node given its set of parent object nodes. In an advanced IR system, the representation nodes are derived by probabilistic indexing, and the nodes are associated with index terms extracted from the document texts. The query node corresponds to the event that an information need is met and multiple roots that correspond to the concepts that express the information need. A set of intermediate query nodes may be used to describe complex query networks such as those formed with Boolean expressions [6].

By structuring the inference net in this way, objects can be retrieved independently of each other. We could, for example, specify that only objects from the class representing single paragraphs be retrieved. In that case, each o_i node that corresponds to a paragraph object is asserted to be *true* (one at a time), and the objects are ranked according to their probability of satisfying the information need.

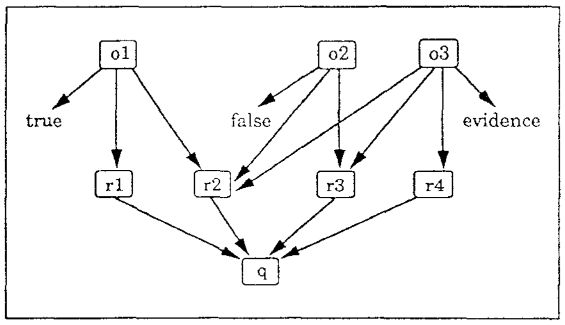


Figure 2: A model for composite object retrieval

In many cases, we want the retrieval of objects to be influenced by the content of subobjects or related objects. For example, in Figure 1, object o_1 has been instantiated. All remaining objects in the network are set to *false*, except those objects that are subobjects of o_1 (o_3 in this case). We assume that all subobjects (i.e. the transitive closure) have evidence attached to them in this way, and that each subobject is set to *true* (the strongest form of evidence). This means that when an object occurs, all of its subobjects occur, with certainty, at the same time. The effect of this evidence is to raise the probability (or belief) associated with all representation concepts that describe subobjects. This effectively adds concepts to o_1's representation and reinforces belief in concepts that are used to describe both the parent object and subobjects. In the example, the addition of o_3 adds concepts r_3 and r_4 to o_1's representation, strengthens belief in r_2 given o_1, and leaves belief in r_1 unchanged.

A similar approach can be used for retrieval of objects which have no text content using related objects. For example, suppose that a user wants to retrieve all figures that satisfy a particular information need. The figure may have a caption, in which case this could be used to retrieve them independently of other objects. In the case where there is no caption, or when the caption is not sufficiently informative, we can instantiate objects that represent other parts of the article, and this instantiation can be done in progressively wider contexts. We could initially instantiate objects representing paragraphs in the same section of the document as the figure. If this did not produce satisfactory results, we could widen the context until the whole document is being used to determine the "meaning" of the figure. Note that this instantiation of objects is going in the opposite direction to the instantiation of subobjects for composite object retrieval, and the evidence attached to objects from wider contexts may be weaker than that attached to objects that are very "close" to the figure.

The disadvantages of this approach to complex object retrieval are the lack of flexibility and computational efficiency. A similar form of the inference net was used for citation experiments [19] and was implemented using inverted files. To do this, however, beliefs had to be computed for each group of objects (documents) that were going to be instantiated at the same time. In the case of complex object retrieval, identifying all these groups prior to searching may be difficult.

A simpler approach to complex object retrieval is based on treating each instantiated object as a separate piece of evidence, and then combining the beliefs associated with this evidence at query time. In other words, each subobject (or related object) is instantiated one at a time, as is done in typical inference net retrieval. The result of this is a set of belief values $P(I|o_1)$, $P(I|o_3)$, etc. These belief values can then be combined to give an overall belief based on all evidence $P(I|o_1, o_3, \ldots)$. This is shown in Figure 3. The combination of the belief values can be done in a variety of ways, as is the case in a query net. For example, objects could be ranked according to the maximum or average belief associated with its subobjects.

The integrated architecture described in this paper uses this second approach. Based on some related retrieval experiments, we compute belief values using the maximum belief of subobjects. In addition, we combine this belief value with a belief value computed using the "whole" object. To do this, object nodes are created by taking the union of all text in the subobjects. In the case of the dissertation example, sections could be retrieved by the belief calculated using all the text in the section, by the maximum belief associated with a paragraph in the section, or by a combination of the two. The belief computed using the entire text content will be very similar to the belief calculated using the first approach shown in Figure 2. Preliminary retrieval experiments show that combining beliefs in this way leads to significant effectiveness improvements, but more work needs to be done. This approach is related to that described in [16].

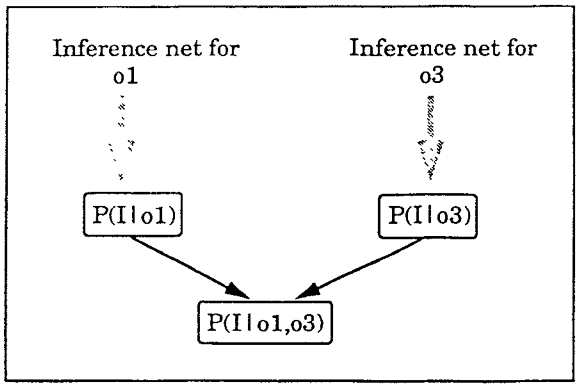


Figure 3: Combining beliefs from subobjects

3 The Document Model and IRIS

In the prototype system described in this paper, dissertations are stored as complex objects in a database system (IRIS), and retrieval is performed using both the database system and a text retrieval system. The model of dissertations used in the database schema describes complex objects similar to that shown in Figure 1.

IRIS is an object-oriented database management system which was developed at Hewlett-Packard Laboratories. The version of IRIS that we are currently using is a research prototype (DPP 4.0 [9]).

The IRIS system is based on a semantic data model which contains support for abstract data types (ADTs). The data model is based on the following three constructs: objects, types, and functions. It supports inheritance, constraints, non-normalized data, user-defined functions, version control, and extensible data types [21, 9].

The objects within the IRIS system can be retrieved (referred to) independently of their attribute values by using a unique object identifier, or OID, that is assigned when the object is created. All objects are classified by type, and are associated with a specific set of functions. Types are organized into a type graph which supports inheritance.

Functions in the IRIS system are used to model all attributes, relationships, and other operations on objects. Functions are inherited by subtypes.

Figure 4 illustrates the architecture of IRIS. The IRIS Kernel implements the IRIS data model described above. There are a number of interfaces, all of which are built as clients of the Kernel, including the follow-

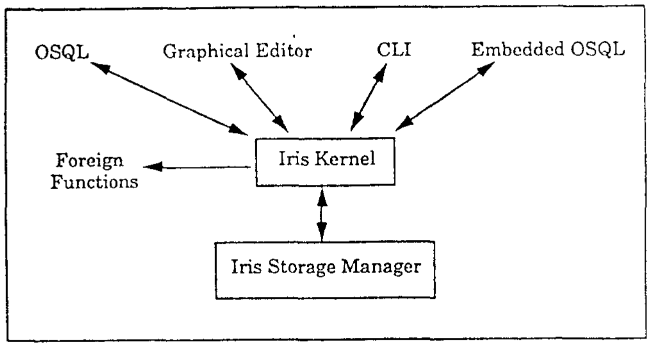


Figure 4: The IRIS Architecture

ing: an object version of SQL (OSQL), the C Language Interface (CLI), a Graphical Editor, and more recently, the IRIS Programming Language (IPL) [1]. The Graphical Editor is an X Windows-based program which allows both retrieval and update of both function values and metadata with graphical and forms-based displays. The C Language Interface allows access to IRIS in an object-oriented fashion through manipulation of C variables which denote the IRIS database, metadata, and objects in the database. The Embedded OSQL interface is also a programming interface, which allows OSQL to be embedded into various host languages.

Foreign functions provide alternative methods for computing function values, and are implemented as subroutines written in some general-purpose programming language and compiled outside of IRIS.

The IRIS Storage Manager provides concurrency control, recovery, buffering, indexing, clustering, and OID generation.

Within the context of the IRIS data model, the schema that was defined to model dissertations contains a number of type definitions, both atomic and complex. In addition, functions (based on transitive closure) are used for determining all subobjects of a particular type (e.g. `chaps()` for chapters of a dissertation, `paras()` for paragraphs of any non atomic type, etc.).

The types described in the schema are:

- Atomic Types (no subobjects): Equation, Paragraph, Picture, Tabular
- Non Atomic Types: Figure, Table, Chapter, Document, Dissertation (subtype of Document), Section, Subsection, Subsubsection, Subsubsubsection,

4 The INQUERY System

INQUERY is a text retrieval system based on the inference net retrieval model [4]. It consists of an indexing module, an interface module, and a text retrieval engine. The indexing module is used to parse the input text, do automatic indexing, and build the associated dictionaries and inverted files. The text retrieval engine uses the inverted files and other data to evaluate query nets described with a query language. The query language can be used to represent complex combinations of concepts. The interface module is currently very simple since INQUERY is primarily designed as a retrieval engine that can be integrated with other systems and interfaces. The current interface accepts simple natural language queries or queries expressed in the INQUERY query language. The indexing and retrieval engine functionality can be accessed using a simple application programmer's interface (API).

The INQUERY system is designed for both experimental (batch) and interactive use. It has been used in other experiments with test collections up to 500 MBytes and is being modified to work with even larger databases. The system is implemented in C and runs on a variety of workstations using UNIX.

5 The Integrated System

The integrated system (COINS) uses the functionality of INQUERY and IRIS to model and store complex objects and retrieve them using their text content. COINS accesses the INQUERY database through a set of C functions, and accesses and manipulates the data/text stored in the IRIS database through C variables.

The database used for the prototype system consists of dissertations written in $\LaTeX$, which have a well-defined, complex structure. The $\LaTeX$ source for the dissertations is scanned, parsed, and used to create database objects in IRIS, and text objects in INQUERY.

The COINS interface allows a user to enter queries that specify the types of objects to be retrieved, the type of retrieval model to use, and the desired contents of the objects. Queries in OSQL and the INQUERY query language are also accepted. Objects are retrieved as ranked lists.

5.1 Indexing the Dissertations

Figure 5 shows the process by which dissertations in $\LaTeX$ are entered into the IRIS and INQUERY databases.

The first step uses `lex` to define and recognize the valid tokens within the source file, and `yacc` to perform actions based on the type of tokens found. Eighteen `lex` regular expressions and fifty-one `yacc` production rules are used to recognize the components of the document structure in the LaTeX source, store objects in the IRIS database, and create a text file for input to INQUERY. The parser uses a history stack to keep track of the hierarchical document structure and to record OIDs for objects as they are stored in IRIS. These OIDs and transitive closure functions are used to output text records to a temporary file. The text record for a section, for example, consists of the OID for the section and all the text in the section (including all subsections, etc.). For a paragraph, the text record consists of the OID for the paragraph and the text of the paragraph.

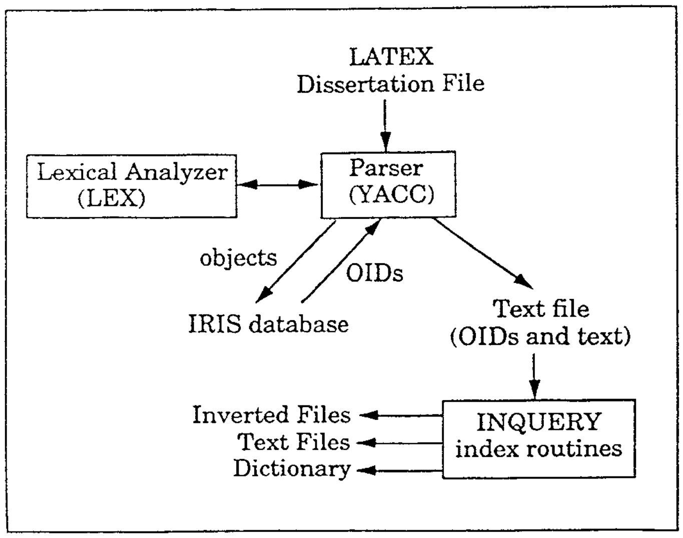


Figure 5: Processing the LaTeX Source

The file of text records is passed to the INQUERY indexing module which creates the appropriate indexes and dictionaries for word- and phrase-based retrieval using the inference net.

The test database contains two dissertations which were parsed into 3,378 objects. Of these, 2,169 were stored in IRIS (some objects such as references and citations were ignored), and 1,950 corresponding text objects were stored in INQUERY (objects such as equation and eqnarray do not have text). A total of 3,438 indexes for terms were created in INQUERY. The size of the dissertation files was approximately 1 MByte, the size of the IRIS files (including system overhead) was approximately 4 MBytes, and the size of the INQUERY indexes was 2.5 MBytes (proximity and other information is currently stored uncompressed).

5.1.1 Retrieval Architecture

The COINS retrieval architecture (Figure 6) supports the retrieval model in Figure 3. That is, it allows retrieval from any level of a a hierarchical object structure, as well as supporting retrieval based on subobjects or related objects. In particular, retrieval can be based on the text of each object only (*normal* mode), based on the text of parent objects (*super* mode), and based on the text of subobjects (*sub* mode). Retrieval based on combinations of these modes is also possible.

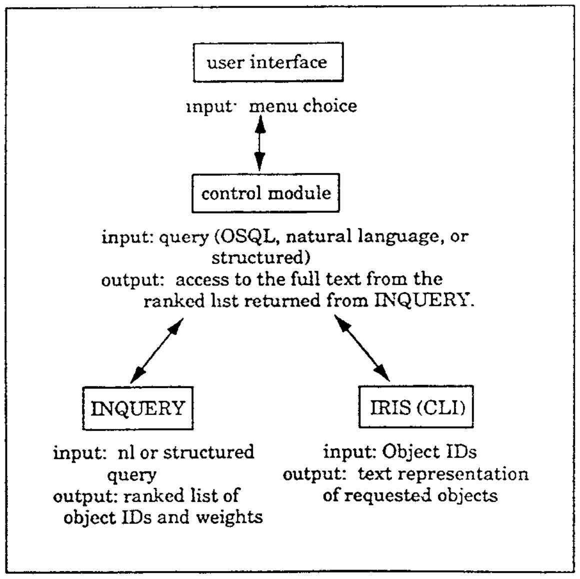

user interface
input: menu choice
control module
input: query (OSQL, natural language, or structured)
output: access to the full text from the ranked list returned from INQUERY.
INQUERY
IRIS (CLI)
input: nl or structured query
output: ranked list of object IDs and weights
input: Object IDs
output: text representation of requested objects

Figure 6: The COINS Retrieval Architecture

The control module is the central module of the COINS retrieval system, and takes the role of the interface between INQUERY and IRIS, as well as between the user interface code and the rest of the system. This module takes as input the query which has been entered into the user interface, and processes it according to the query type. If an OSQL query is entered, the control module simply calls IRIS to return the results of that query. If a natural language request or query in the INQUERY language has been entered, the control module makes a function call to INQUERY. INQUERY returns a ranked list of objects with their associated probabilities of satisfying the information need (scores) and their OIDs. The control module then filters the ranked list based on the specified object type and retrieval mode. The OIDs are used to access objects in IRIS as part of the filtering process.

Figure 7 shows an example of searching in the COINS system. The first screen shows the specification to retrieve subsections using the *normal* retrieval mode. The "amount of text to retrieve" specifies that full text is required for browsing, as opposed to a short summary. The second screen shows the query, the initial number of objects retrieved from INQUERY, and the number of objects after they have been filtered by object type. The third screen shows the ranked list of objects retrieved by INQUERY (including the OIDs, which are not intended for end user display). The fourth screen shows the text of the top-ranked subsection, with query words highlighted.

Figure 8 shows the results of retrieving chapters using the *sub* retrieval mode based on paragraphs. In other words, the rank of the chapter is determined by the maximum probability value associated with a paragraph in the chapter. In this case, many more objects are retrieved from INQUERY than are eventually displayed after filtering.

Finally, Figure 9 shows the results of retrieving figures based on the *super* mode using subsections. The query was to retrieve figures that involve a comparison of the inference network model to Fuhr's model. The caption of the top ranked figure, displayed in the second screen, does not contain a critical word in the query (the caption is "inference network for the RPI model" and the RPI model is another name for Fuhr's model). Because retrieval was based on the text of the subsection that contained the figure, it is not necessary to rely on the limited caption text.

6 Implementation Issues

The version of IRIS that was used for COINS had a number of limitations that forced us into particular design decisions. Most of these limitations have not had an affect on the functionality of COINS, but complicated the implementation of the system.

Initially, we planned to use the foreign function capability to integrate the INQUERY functionality with IRIS. The foreign functions would simply be INQUERY functions and the entire application could have been written in OSQL, with some links to external code segments. The problem with that approach was that foreign functions are not available in the prototype version of IRIS. In a future version of IRIS, it would be of interest to examine this approach since it would result in a more integrated query language.

In the prototype system, the object size limit is 4K, which includes some overhead for the system, so the maximum amount of data that we are able to store within a particular object is 3996 bytes. When dealing with text, this is not very large, and we were only able to store text within IRIS at the lowest object level in which it occurred. In this schema, that is the paragraph level. The database then is created by storing text with the object at the lowest level, and creating links between that level and the immediate parent object. The full text of an object such as a section can be retrieved using the transitive closure function provided in IRIS.

Finally, it should be noted that the prototype system is very slow. Using IRIS to filter the output of INQUERY is a time-consuming operation, and for efficiency reasons, additional indexes were created using INQUERY to provide information about object type. The process of creating the database is also unacceptably slow (more than 10 hours for the 2000 objects in the two dissertations).

7 Conclusion

The COINS system is a complete implementation of a complex object retrieval model based on inference nets. The loosely-coupled integration of an object-oriented database system and a text retrieval system was shown to be a feasible platform for this retrieval model, and a variety of queries based on the structure and content of the objects that make up long, complex documents can be processed. The implementation highlighted a number of problems with the loosely-coupled architecture, primarily related to efficiency. This system, however, should only be regarded as a step towards the eventual goal of an integrated database/text retrieval system for complex objects.

Our main emphasis in future research will be to investigate tightly-coupled integrations. In particular, we are interested in developing probabilistic object algebras, similar to the probabilistic relational algebras discussed in [11]. An algebra of this would support the manipulation of object structure and probabilities produced by the underlying retrieval algorithms.

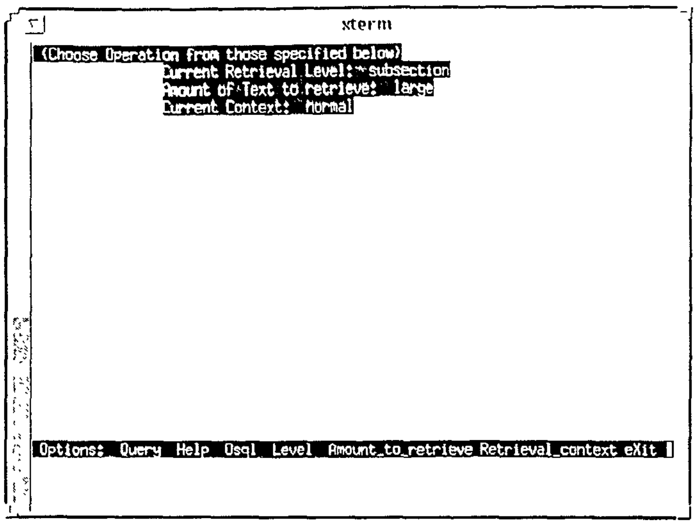
Acknowledgments

This research was supported in part by AFOSR contract 91-0324, Hewlett Packard Corporation, and Digital Equipment Corporation.

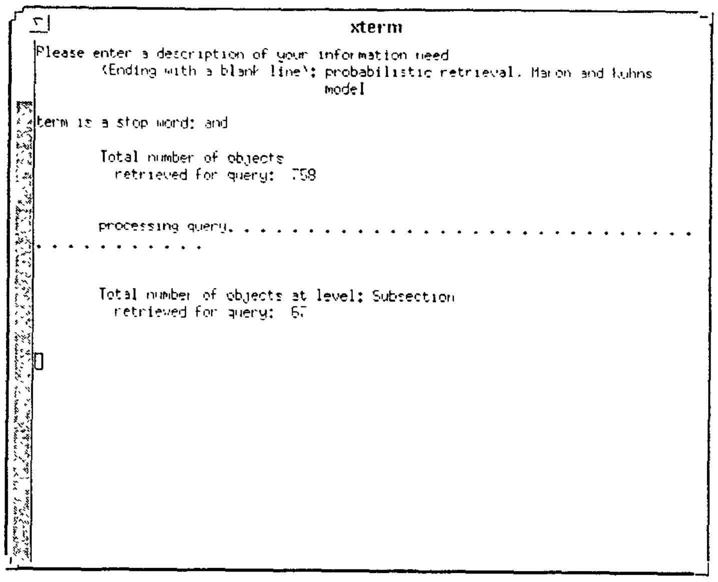
References

[1] J. Annevelink. Database programming languages: A functional approach. In *1991 ACM SIGMOD International Conference on Management of Data*, pages 318–327, 1991.

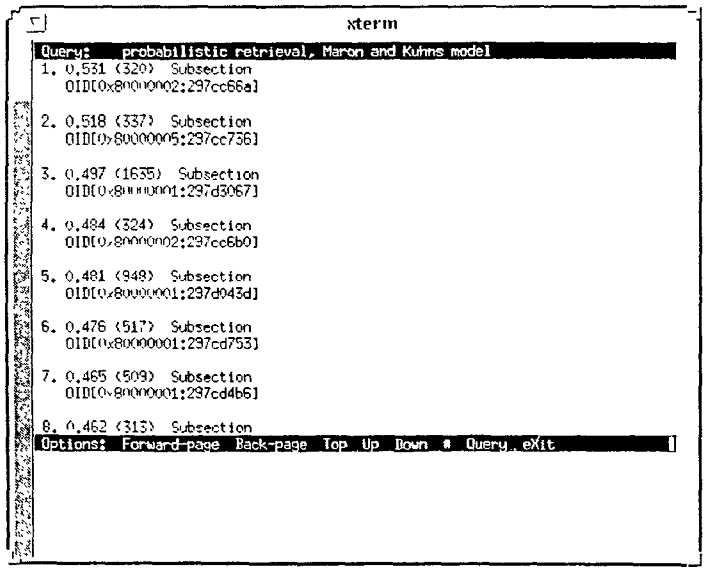
[2] J. Banerjee, H. Chou, J.F. Garza, W. Kim, D. Woelk, and N. Ballou. Data model issues for object-oriented applications. *ACM Transactions on Office Information Systems*, 5(1):3–26, 1987.



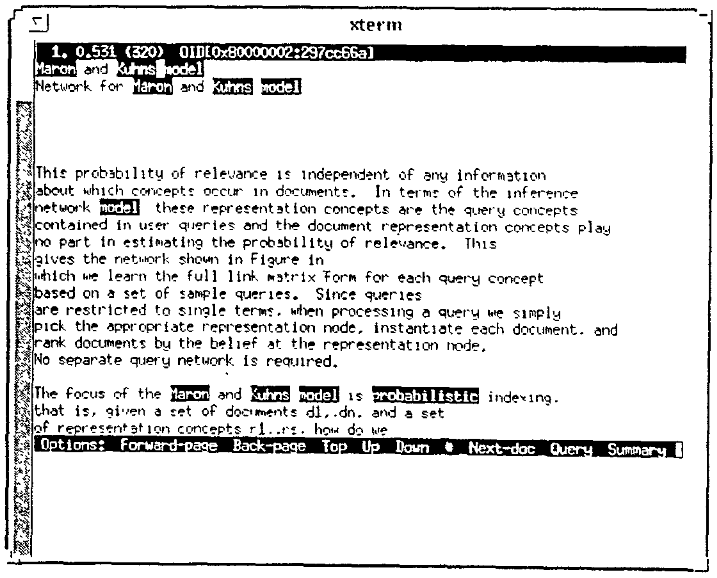
```
(Choose Operation from those specified below)
                  Current Retrieval Level: subsection
                  Amount of Text to retrieve: large
                  Current Context: normal

Options: Query Help Osql Level Amount_to_retrieve Retrieval_context eXit
```



```
Please enter a description of your information need
        (Ending with a blank line): probabilistic retrieval. Maron and Kuhns
                                    model

term is a stop word: and

        Total number of objects
          retrieved for query:  758


        processing query. . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . .


        Total number of objects at level: Subsection
          retrieved for query:  67
```

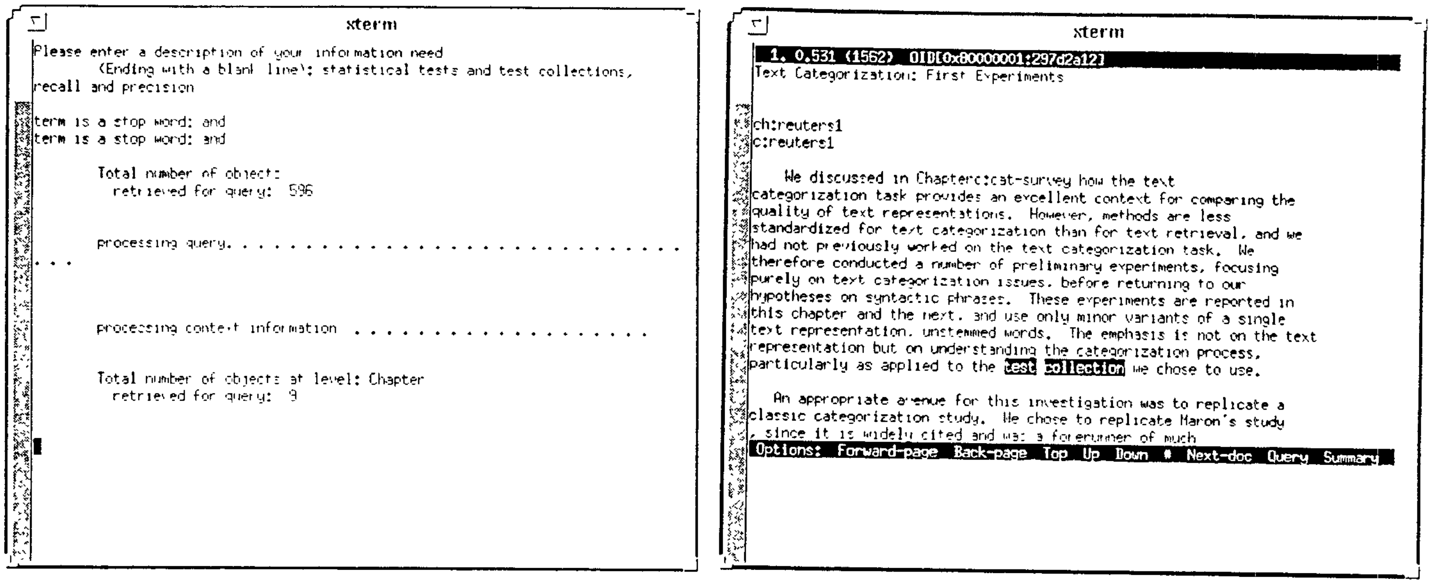


```
Query:    probabilistic retrieval, Maron and Kuhns model
1. 0.531 (320)  Subsection
   OID[0x80000002:297cc66a]

2. 0.518 (337)  Subsection
   OID[0x80000005:297cc736]

3. 0.497 (1635)  Subsection
   OID[0x80000001:297d3067]

4. 0.484 (324)  Subsection
   OID[0x80000002:297cc6b0]

5. 0.481 (948)  Subsection
   OID[0x80000001:297d043d]

6. 0.476 (517)  Subsection
   OID[0x80000001:297cd753]

7. 0.465 (509)  Subsection
   OID[0x80000001:297cd4b6]

8. 0.462 (313)  Subsection
Options: Forward-page Back-page Top Up Down # Query eXit
```



```
 1. 0.531 (320)  OID[0x80000002:297cc66a]
Maron and Kuhns model
Network for Maron and Kuhns model



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

The focus of the Maron and Kuhns model is probabilistic indexing.
that is, given a set of documents d1,.dn. and a set
of representation concepts r1..rs. how do we
Options: Forward-page Back-page Top Up Down # Next-doc Query Summary
```

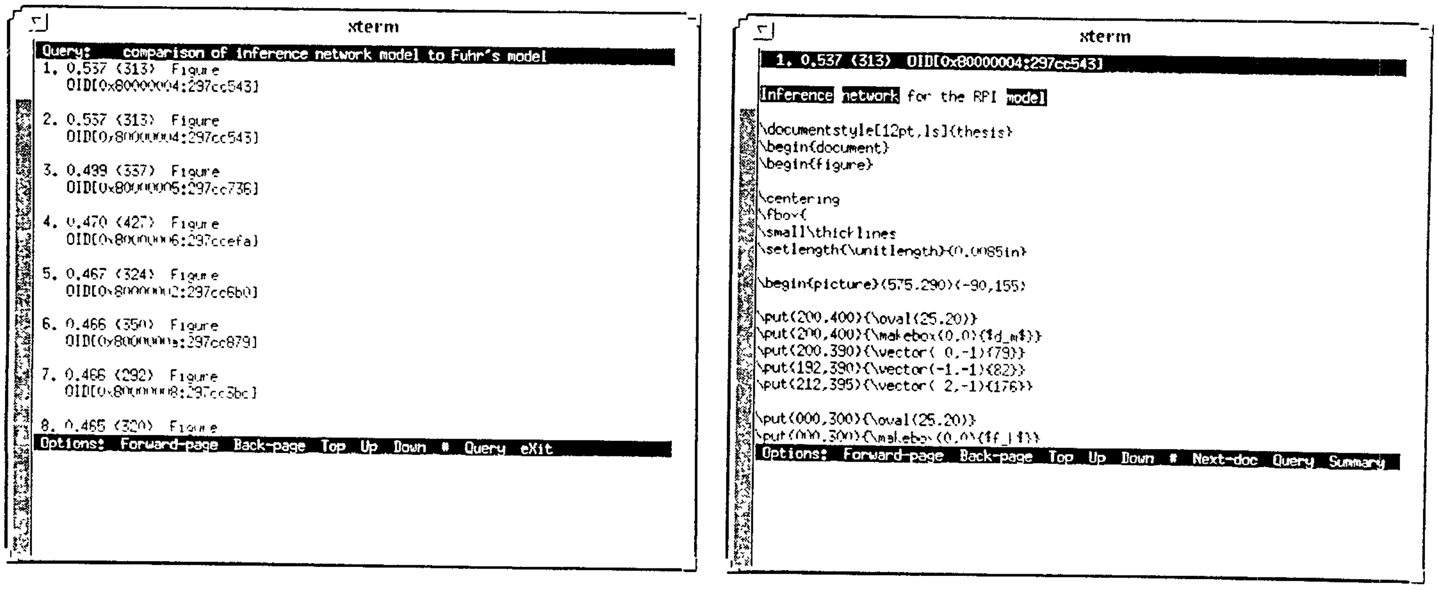
Figure 7: An example of *normal* mode retrieval



```
xterm
Please enter a description of your information need
        (Ending with a blank line): statistical tests and test collections,
recall and precision

term is a stop word: and
term is a stop word: and

        Total number of object:
          retrieved for query:  596


        processing query. . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . .



        processing context information  . . . . . . . . . . . . . . . . . . .


        Total number of objects at level: Chapter
          retrieved for query:  9
```

```
xterm
 1. 0.531 (1562)  OID[0x80000001:297d2a12]
Text Categorization: First Experiments


ch:reuters1
c:reuters1

    We discussed in Chapterc:cat-survey how the text
categorization task provides an excellent context for comparing the
quality of text representations.  However, methods are less
standardized for text categorization than for text retrieval, and we
had not previously worked on the text categorization task.  We
therefore conducted a number of preliminary experiments, focusing
purely on text categorization issues, before returning to our
hypotheses on syntactic phrases.  These experiments are reported in
this chapter and the next, and use only minor variants of a single
text representation, unstemmed words.  The emphasis is not on the text
representation but on understanding the categorization process,
particularly as applied to the test collection we chose to use.

   An appropriate avenue for this investigation was to replicate a
classic categorization study.  We chose to replicate Maron's study
, since it is widely cited and was a forerunner of much
Options:  Forward-page  Back-page  Top  Up  Down  #  Next-doc  Query  Summary
```

Figure 8: An example of *sub* mode retrieval



```
xterm
Query:    comparison of inference network model to Fuhr's model
 1. 0.537 (313)  Figure
    OID[0x80000004:297cc543]

 2. 0.537 (313)  Figure
    OID[0x80000004:297cc543]

 3. 0.499 (337)  Figure
    OID[0x80000005:297cc736]

 4. 0.470 (427)  Figure
    OID[0x80000006:297ccefa]

 5. 0.467 (324)  Figure
    OID[0x80000002:297cc6b0]

 6. 0.466 (350)  Figure
    OID[0x80000003:297cc879]

 7. 0.466 (292)  Figure
    OID[0x80000008:297cc3bc]

 8. 0.465 (320)  Figure
Options:  Forward-page  Back-page  Top  Up  Down  #  Query  eXit
```

```
xterm
 1. 0.537 (313)  OID[0x80000004:297cc543]

Inference network for the RPI model

\documentstyle[12pt,ls]{thesis}
\begin{document}
\begin{figure}

\centering
\fbox{
\small\thicklines
\setlength{\unitlength}{0.0085in}

\begin{picture}(575.290)(-90,155)

\put(200.400){\oval(25.20)}
\put(200,400){\makebox(0.0){$d_m$}}
\put(200.390){\vector( 0,-1){79}}
\put(192,390){\vector(-1.-1){82}}
\put(212,395){\vector( 2,-1){176}}

\put(000,300){\oval(25.20)}
\put(000,300){\makebox(0.0){$f_1$}}
Options:  Forward-page  Back-page  Top  Up  Down  #  Next-doc  Query  Summary
```

Figure 9: An example of *super* mode retrieval

[3] David C. Blair. An extended relational retrieval model. *Information Processing and Management*, 24(3):349–371, 1988.

[4] J. P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. Technical report, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, 1992.

[5] M.P. Consens and A.O. Mendelzon. Expressing structural hypertext queries in graphlog. In *Proceedings of Hypertext 89*, pages 269–292, 1989.

[6] W. B. Croft, H.R. Turtle, and D.D. Lewis. The use of phrases and structured queries in information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 32–45, 1991.

[7] W. Bruce Croft, R. Krovetz, and H. R. Turtle. Interactive retrieval of complex documents. *Information Processing and Management*, 26(5):593–613, 1990.

[8] W. Bruce Croft and Howard Turtle. Retrieval of complex objects. In *Proceedings of EDBT 92*, 1991. (to appear).

[9] D.H. Fishman. Overview of the Iris dbms. *Hewlett Packard Technical Report*, HPL-SAL-89-15, 1989.

[10] Norbert Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of VLDB 90*, pages 696–707, 1990.

[11] H. Garcia-Molina and D. Porter. Supporting probabilistic data in a relational system. In *Proceedings of EDBT*, pages 60–74, 1990.

[12] C. A. Lynch and M. Stonebraker. Extended user-defined indexing with applications to textual databases. In *Proceedings of the Very Large Database Conference*, pages 306–317, 1988.

[13] I.A. Macleod and R.G. Crawford. Document retrieval as a database application. *Information Technology: Research and Development*, 2:43–60, 1983.

[14] D. Maier and J. Stein. Development and implementation of an object-oriented dbms. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 355–392. MIT Press, 1987.

[15] Amihai Motro. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions of Office Information Systems*, 6(3):187–214, July 1988.

[16] Gerard Salton and Chris Buckley. Global text matching for information retrieval. *Science*, 253:1012–1015, 1991.

[17] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[18] H.J. Schek. Methods for the administration of textual data in database systems. In C.J. Van Rijsbergen, R.N. Oddy, and P.W. Williams, editors, *Research and Development in Information Retrieval*, pages 218–235, 1981.

[19] Howard R. Turtle. *Inference Networks for Document Retrieval.* PhD thesis, University of Massachusetts at Amherst, 1990.

[20] 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.

[21] K. Wilkinson, P. Lyngbaek, and W. Hasan. The Iris architecture and implementation. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):63–75, 1990.

[22] S. B. Zdonik and D. Maier. *Readings in Object-Oriented Database Systems.* Morgan Kaufmann, San Mateo, CA, 1990.