Spring 5-31-2001

Knowledge-based document retrieval with application to TEXPROS

Fang Sheng
New Jersey Institute of Technology

ABSTRACT

KNOWLEDGE-BASED DOCUMENT RETRIEVAL WITH APPLICATION TO TEXPROS

By
Fang Sheng

Document retrieval in an information system is most often accomplished through keyword search. The common technique behind keyword search is indexing. The major drawback of such a search technique is its lack of effectiveness and accuracy. It is very common in a typical keyword search over the Internet to identify hundreds or even thousands of records as the potentially desired records. However, often few of them are relevant to users' interests.

This dissertation presents a knowledge-based document retrieval architecture with application to TEXPROS. The architecture is based on a dual document model that consists of a document type hierarchy and a folder organization. Using the knowledge collected during document filing, the search space can be narrowed down significantly. Combining the classical text-based retrieval methods with the knowledge-based retrieval can improve tremendously both search efficiency and effectiveness.

With the proposed predicate-based query language, users can more precisely and accurately specify the search criteria and their knowledge about the documents to be retrieved. To assist users formulate a query, a guided search is presented as part of an intelligent user interface. Supported by an intelligent question generator, an inference engine, a question base, and a predicate-based query composer, the guided search collects the most important information known to the user to retrieve the documents that satisfy users' particular interests.

A knowledge-based query processing and search engine is presented as the core component in this architecture. Algorithms are developed for the search engine to effectively and efficiently retrieve the documents that match the query.

Cache is introduced to speed up the process of query refinement. Theoretical proof and performance analysis are performed to prove the efficiency and effectiveness of this knowledge-based document retrieval approach.

KNOWLEDGE-BASED DOCUMENT RETRIEVAL WITH APPLICATION TO TEXPROS

by
Fang Sheng

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer and Information Science

Department of Computer and Information Science

May 2001

APPROVAL PAGE

KNOWLEDGE-BASED DOCUMENT RETRIEVAL WITH APPLICATION TO TEXPROS

Fang Sheng

Dr. Gary Thomas, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT, Newark, NJ

Dr. Peter A. Ng, Dissertation Co-Advisor — Date
Professor of Computer Science, University of Nebraska at Omaha, Omaha, NE

Dr. D.C. Douglas Hung, Committee Member — Date
Associate Professor of Computer and Information Science, NJIT, Newark, NJ

Dr. Ajaz Rana, Committee Member — Date
Assistant Professor of Computer and Information Science, NJIT, Newark, NJ

Dr. Ronald S. Curtis, Committee Member — Date
Associate Professor of Computer Science, William Paterson University, Wayne, NJ

BIOGRAPHICAL SKETCH

Author: Fang Sheng

Degree: Doctor of Philosophy

Date: May 2001

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer and Information Science,
  New Jersey Institute of Technology, Newark, New Jersey, 2001
- Master of Computer Science,
  New Jersey Institute of Technology, Newark, New Jersey, 1998
- Master of Computer Science and Engineering,
  Tsinghua University, Beijing, China, 1991
- Bachelor of Computer Science and Engineering,
  Tsinghua University, Beijing, China, 1988

Major: Computer Science

Presentations and Publications:

Fang Sheng, Gary Thomas and Peter A. Ng, "Intelligent Document Retrieval: A Knowledge-Based Approach", To appear in Proceedings of the Fifth World Multi-conference on Systemics, Cybernetics and Informatics, July 2001.

Xien Fan, Fang Sheng, Xuhong Li, Zhenfu Cheng and Peter A. Ng, "A Scalable Automated System for Document Management", In Proceedings of the Fifth World Conference on Integrated Design and Process Technology, June 2000.

Xuhong Li, Zhenfu Cheng, Fang Sheng, Xien Fan and Peter A. Ng, "A Document Classification and Extraction System with Learning Ability", in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, June 2000.

Xien Fan, Fang Sheng and Peter A. Ng, "DOCPROS: A Knowledge-Based Personal Document Management System", In Proceedings of the 10th International Workshop on Database and Expert Systems Applications, pp. 527-531, September 1999.

Xien Fan, Fang Sheng, Simon Doong, Peter A. Ng and Ching-Song Don Wei, "A Process for Constructing a Personal Folder Organization", in Proceedings of the International Workshop on Multimedia Database, pp. 20 – 27, August 1998.

This dissertation is dedicated to
my son
and
my husband

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor, Professor Gary Thomas, for his valuable guidance, comments, and encouragement to make this final manuscript possible. Also, I would like to express my deep appreciation to my co-advisor, Professor Peter A. Ng, for his long time support, guidance and constant encouragement through the whole process. Special thanks are given to Dr. D.C. Douglas Hung, Dr. Ajaz Rana and Dr. Ronald S. Curtis for actively participating in my committee.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Today, a typical office generates hundreds of thousands of documents. With the boom of Internet technologies, documents in electronic formats have increased dramatically. Such a phenomenon has a huge impact on offices of the government, businesses and home. Efficient and accurate document storage and retrieval are becoming more imposing and difficult. Researchers from all over the world have been investigating various aspects of document automation to simplify the process of tedious document processing and to improve the office work environment. Many document processing systems and methodologies have been developed. This chapter will introduce briefly some of the related research work.

1.1 Overview of Related Work

Some information storage and retrieval research is focused on document image retrieval. Printed documents including newspapers and business letters are often scanned (for archiving or in an attempt to move toward a paperless office) and stored as images. In order to make full use of the capabilities of traditional database indexing and retrieval techniques, a full conversion of the document may be required. There are many factors, however that may prohibit complete conversion -- including its high cost, insufficient document quality, or the fact that parts of the document simply cannot be adequately represented in a converted format. Methods have been developed to access document images without relying on complete and accurate conversion.

For example, J. F. Cullen, J. J. Hull and P. E. Hart [11] built a system that uses texture to retrieve and browse images stored in a large document image database. A method of graphically generating a candidate search image is used that shows the visual layout and content of a target document. All images similar to this candidate are returned

so that they may be browsed or for further inquiry. The system is accessed using a Web browser.

A. F. Smeaton and A. L. Spitz [49] developed a method that makes generalizations concerning the images of characters first, and then performs classification and agglomerates the resulting character shape codes into word tokens based on character shape coding. They are specific in their representation of the underlying words to allow reasonable performance of retrieval.

Y. He, Z. Jiang, B. Liu and H. Zhao [22] proposed an index method based on stroke density code to retrieve Chinese document images. This method firstly segments the document image to get all the Chinese character images, then calculates stroke density of each Chinese character image, and finally, obtains a stroke density code of the character image. The index method has the advantage of speed and robustness to noise.

B. W. Stalcup, P. W. Dennis and R. B. Dydyk[51] developed the Imaged Document Optical Correlation and Conversion System (IDOCCS) that provides a total solution to the problem of managing and retrieving textual and graphic information from imaged document archives. IDOCCS can be used to rapidly search for keywords or phrases within the imaged document archives, as well as to automatically compare an input document with the archived database to determine if it is a duplicate.

A few systems and methods have been developed to process multimedia documents including audio, video and images. STRETCH [2] developed a system for storing and retrieving imaged multimedia document by content. The core of STRETCH system is a powerful archiving and retrieval Engine, based on a structured document representation and capable of activating appropriate methods to characterize and automatically index heterogeneous documents with variable layout and subsequently retrieve them by answering complex queries.

ToCAI [1] proposed a framework for indexing and retrieval of multimedia documents, which presents the ToCAI (Table of Content-Analytical Index) description

scheme for content description of audio-visual documents. The original idea comes from the structure used for technical books. One may easily understand a book sequential organization by looking at its table of contents to quickly retrieve elements of interest by means of the analytical index. This description scheme provides a hierarchical description of the time sequential structure of a multimedia document, suitable for browsing, together with an "Analytical Index" (AI) of audio-visual objects of the document, suitable for effective retrieval.

L. Wilcox and J. Boreczky [56] proposed a method for indexing and retrieval of multimedia audio and video data based on annotation and segmentation. Annotation refers to the association of text data with particular time locations of the media. Segmentation is the partitioning of continuous media into homogenous regions. Retrieval is performed over segments of the media using the annotations associated with the segments.

W. W. Chu, C. C. Hsu, A. F. Cardenas, and R. K. Taira [10] developed a knowledge-based approach to retrieve medical images by feature and content with spatial and temporal constructs. Selected objects of interest in a medical image (e.g., x-ray, MR image) are segmented, and contours are generated from these objects. Features (e.g., shape, size, texture) and content (e.g., spatial relationships among objects) are extracted and stored in a feature and content database. Knowledge about image features are expressed as a hierarchical structure called a Type Abstraction Hierarchy (TAH). A knowledge-based spatial temporal query language (KSTL) was developed to support approximate matching of feature and content, conceptual terms, and temporal logic predicates.

P. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel and Z. Protopapas[27] developed a method to retrieve medical tumors that are similar to a given pattern from a large medical database. It used a natural similarity function for shape-matching, based on concepts from mathematical morphology, and it can be lower-bounded by a set of shape

features for safely pruning candidates, thus giving fast and correct output. These features were organized in a spatial access method, leading to fast indexing for range queries and nearest-neighbor queries.

E. Ozkarahan [45] proposed an integrated conceptual representation scheme for multimedia documents. It developed the necessary abstractions for the conceptual model and extensions to the RM/T relational model used as the search structure. It then developed a retrieval model in which the database search space is first narrowed down, based on user query, by an associative search. The associative search is followed by semantic and media-specific searches. A query language called SQLX is introduced to formulate these searches directly from the conceptual model.

Much research work has been contributed to text-based document retrieval. CONCERTO [4] is a comprehensive document processing system that supports indexing, querying and retrieval of digital documents. A. Celentano, M. Fugini, and S. Pozzi[5] proposed a distributed client/server architecture that supports the classification, filing and retrieval of documents and the maintenance of system knowledge.

P. O'Neil [43] proposed a method that provides for an automatic representation of text data by vectors that can then be manipulated to categorize and organize data. Information can be retrieved without knowledge of the underlying process. The user can ask for information using normal discourse.

R. M. Rohrer, J. L. Sibert and D. S. Ebert [42] proposed a shape-based visual interface for text retrieval and interactive exploration. The exploratory system uses procedurally generated shapes coupled with an underlying text retrieval engine. Traditional text-based queries and summarization are enhanced with a visual interface based on 3D shapes (glyphs). This interface allows visualizing multidimensional relationships among documents and perceiving more information than with conventional text-based interfaces.

R. Baeza-Yates and G. Navarro [3] focused on the issue of reducing the space overhead when indexing large text database as the text collections grow in size. They studied the space overhead and retrieval times as functions of the block size and found that, under reasonable assumptions, it is possible to build an index that is simultaneously sub-linear in space overhead and in query time.

R. Marega and M. T. Pazienza [38] developed an information retrieval using semantic information, which can be automatically acquired by applying natural language processing (NLP) techniques to texts. The information is represented using conceptual graphs. The problem of synonyms and homonyms is addressed in the system by using a model based on the interpretation of conceptual graphs extracted from texts. The detection of contextual roles of words allows an improvement in retrieval precision over traditional IR technologies. Ranking, based on document relevance, is obtained by extending the vector space model into an oblique space and taking into account the relevance among different word couples.

H. Chen [7] presented research on the design of knowledge-based document retrieval systems. A semantic network structure was adopted to represent subject knowledge, classification scheme knowledge, modeled experts' search strategies and user modeling capability as procedural knowledge. These functionalities were incorporated into a prototype knowledge-based retrieval system. This system was able to create a user profile, identify task requirements, suggest heuristics-based search strategies, perform semantic-based search assistance, and assist online query refinement.

Increasing amounts of research work has been done to address specific issues concerned with the World Wide Web. The Web is currently a distributed mass of simple hypertext documents. The languages we currently use to associate metadata with Web resources are insufficient to satisfy all the requirements necessary to support precise, flexible and scalable knowledge representation and information retrieval. P. Martin and P. W. Eklund[39] explored the requirements for metadata languages to support

knowledge representation and retrieval on the Web. They also presented a new tool, WebKB that interprets semantic statements stored in Web-accessible documents.

The World Wide Web is a world of great richness, but finding information on the Web is also a great challenge. In order to clarify the ambiguity of the short queries given by users, C. Chang and C. Hsu [6] proposed the idea of concept-based relevance feedback for Web information retrieval. The idea is to have users give two to three times more feedback in the same amount of time that would be required to give feedback for conventional feedback mechanisms. Under this design principle, they apply clustering techniques to the initial search results to provide concept-based browsing.

M. Ortega-Binderberger, S. Mehrotra, K. Chakrabarti and K. Porkaew [44] focused on using an integrated textual and visual search engine for Web documents. Query refinement is supported to enable cross-media browsing in addition to regular searches.

D. Skuce [52] proposed a prototype system, IKARUS, with the potential of integrating Web-based documents, shared knowledge bases, and information retrieval for improving knowledge storage and retrieval. As an example, this paper discussed how to implement both a user manual and an online help system as one system. The following technologies are combined: a Web-based design, a frame-based knowledge engine, use of an advanced full-text search engine, and simple techniques to control terminology.

A considerable amount of research is focused on applying and incorporating artificial intelligent technologies to document retrieval. Classical methods such as Boolean searches, the vector space model and probabilistic retrieval have been applied to the field of document retrieval. However, these methods cannot handle the increasing demands of end-users in satisfying their needs.

M. Cutler, H. Deng, S. S. Maniccam, W. Meng [12] developed a methodology using the structures and hyperlinks of HTML documents and a genetic algorithm to improve the effectiveness of retrieving HTML documents. This methodology partitions

the occurrences of terms in a document collection into classes according to the tags in which a particular term appears (such as Title, H1-H6, and Anchor). The rationale is that terms appearing in different structures of a document may have different significance in the identification of a document. The weighting schemes of traditional information retrieval were extended to include class importance values. A genetic algorithm was implemented to determine a `best so far' class importance factor combination. The experiments indicate that using this technique the retrieval effectiveness can be improved by 39.6% or higher.

S. C. Hui and A. Goh [26] focused on how to apply neural networks and fuzzy logic for document retrieval. In particular, the Fuzzy Kohonen Neural Network (FKNN) is used as an example to illustrate the versatility of fuzzy neural networks as applied to the document-retrieval process. The issues of training, pattern recall, ranking, relevance feedback and performance issues of the FKNN document-retrieval process are discussed.

S. Chen and Y. Horng [8] developed a new method for fuzzy query processing for document retrieval based on extended fuzzy concept networks. In an extended fuzzy concept network, there are four kinds of fuzzy relationships between concepts, i.e., fuzzy positive association, fuzzy negative association, fuzzy generalization, and fuzzy specialization. An extended fuzzy concept network is modeled as a relational matrix and a relevance matrix, where the elements in a relational matrix represents the fuzzy relationships between concepts, and the elements in a relevance matrix indicate the degrees of relevance between concepts. The implicit fuzzy relationships between concepts are inferred by a transitive closure of the relation matrix. The implicit degrees of relevance between concepts are inferred by a transitive closure of the relevance matrix. This method allows the users to perform fuzzy queries in a more flexible and more intelligent manner.

J. Horng and C. Yeh [23] proposed a novel approach to automatic retrieval of keywords and then the uses of genetic algorithms to adapt the keyword weights.

This approach combines the Bigram model and PAT-tree structure to retrieve any type of keywords, such as technical keywords and proper names. In comparison with the PAT-tree based approach, this approach is more effective and faster. Genetic algorithms are used to tune the weight of retrieved keywords.

1.2 TEXPROS and Previous Work

TEXPROS (TEXt PROcessing System) [37] is a comprehensive document management system that provides a computerized environment for users to manage their personal documents. The major components of TEXPROS include document classification and extraction, document organization and filing, a document browser, and document retrieval.

TEXPROS is built based on the dual-model approach, that makes TEXPROS unique and standout from the other systems. The dual-model approach is a flexible, dynamic office document modeling method that consists of a document type hierarchy and a folder organization. The document type hierarchy is used to capture the layout, and the logical and conceptual structures of documents. In TEXPROS, documents are categorized into different classes. Each document class is represented by a frame template, which describes the common properties of the class and is referred to the document type. The frame templates define how documents should be abstracted and interpreted. In addition to document type hierarchy, a folder organization is used to mimic the real world structure for organizing and storing documents in an office environment. Both document type hierarchy and folder organization is flexible and user-oriented.

Document classification and extraction [29, 30, 31, 54, 55] is designed to find the best-fit document type for a given document (and therefore, its associated frame template is retrieved), and then to instantiate a frame instance of its type (represented by its frame template) by filling in the underlying template with significant information of the

document pertinent to the user. This reduces the document size considerably without any severe loss of content. Storage space and processing time are saved in many applications by using frame instances instead of full documents. In the most recent work [31], an automatic document classification and extraction system is developed to support a complete procedure, which begins with the scanning of the paper document into the system and ends with the output of an effective digital form of the original document. A representation of document layout structure Labeled Directed Weighted Graph (LDWG) and an algorithm of transforming document segmentation into LDWG representation are proposed. To find a match between two LDWGs, string representation matching is applied first instead of doing graph comparison directly, which reduces the time for making the comparison. Applying artificial intelligence, the system is able to learn from experiences and builds samples of LDWGs to represent each document type. This document classification and extraction system is domain independent and can be adapted easily to all kinds of application domains.

Document organization and filing [13, 15, 16, 17, 18, 19, 30, 57, 58] is designed to organize documents in efficient ways and to find automatic approach building the document base. In the most recent work [15, 16, 17, 18, 19], a knowledge-based document filing system is presented. The document filing is predicate-driven process. The user can specify filing criteria in terms of predicates. The two elements of the dual model are incorporated by the three-level storage architecture. This storage architecture supports efficient document and information retrieval by limiting the searches to those frame instances of a document type within those folders that appear to be the most similar to the corresponding queries. A knowledge base is presented to contain the knowledge acquired during the document filing process, which is created and updated by a learning agent.

The document browser [14, 35, 36, 53] provides an interactive Graphical User Interface (GUI), through which users can browse documents stored in the system.

Users are required to fully participate in the browsing process by traversing the document organization and repository from top to bottom. In [53], a three-layer architecture for the document browser was proposed. At the top layer, the browsing process controller conducts and monitors the browsing process and utilizes the services provided by the service providers in the second layer. At the bottom layer, the storage management system stores the document and frame instances and responses to the requests from the service providers in the second layer. An infrastructure OP-Net was developed to transform objects in the object network into the predicate-augmented information repository. The predicate associated with each information repository governs the relevant documents during the browsing process and is updated according to user's query. A ranking model based on the signature of the documents and the user's query was also developed. In [14], some improvements were made based on [53]. The user interface is more friendly by adding functions such as "zoom in" and 'zoom out" as well as help, which give users an easier way to view a large graph in one window and provide users help during browsing. A reusable base that holds the basic components was developed to speed up the retrieval.

The document retrieval subsystem is normally a query language based interface, from which the user can specify and submit a query, and then wait for the search engine to return documents that satisfy the user's search criteria. Document retrieval is a task-oriented process. A user can enter a request to tell the system what he/she wants to retrieve. The system processes the request and returns the result. The searching process and details are totally hidden from the user. In [35, 36], an architecture with the capability of processing incomplete and vague queries was proposed. A simple Structured Query Language (SQL), Select-From-Where was developed for the retrieval. A thesaurus and an object network were proposed to support incomplete and vague queries.

The TEXPROS system architecture at component level can be depicted in Figure. 1.1. At the input stage, upon the arrival of an original document, the document

classification and extraction classifies the document and, based on the document type, generates its corresponding frame instance. Then the original document with its corresponding frame instance are passed to the document organization and filing for storing in their proper storage. That means, both will be stored in the document base. During the phase of document classification and extraction, the document type hierarchy gets updated if a new document type is found, whereas, the folder organization requires to be updated, upon the insertion of the new document and its corresponding frame instance into the file organization, during document organization and filing. To search any documents stored in TEXPROS, users can use either document browser or document retrieval component. The document browser provides users with interactive graphic interfaces for browsing across the document base. Document retrieval provides a language-based interface that allows users to retrieve documents by composing and submitting a query.

1.3 TEXPROS and KABIRIA

This section discusses and makes comparisons between TEXPROS and other comprehensive document management systems.

There are several document management systems that focused on text document filing and retrieval. The recent work is the KABIRIA [5]. It is a knowledge-based document filing and retrieval system that supports the classification, filing, and retrieval of documents and the maintenance of system knowledge. A conceptual document model and a document retrieval model are used to present the document structure and operational meaning. The conceptual document model is used for document classification and description and is concerned with the semantic aspects that are not directly and explicitly contained the document text. Each document is modeled by means of a conceptual structure for describing the semantic properties of the document. The document retrieval model is built on the conceptual document model that attempts to

describe the role of a document in an office and its dependencies in the application domain. The documents are then categorized into different classes. Documents with the same structure, meanings and roles in offices are instances of a particular class. Document classes are organized as an is-a hierarchy defining the refinement and generalization relationships. At the input stage, incoming documents are passed to an acquisition and classification module, where partial tree matching is carried out to identify the possible matches in a given document class. Users must be involved in the identification of the class to which an incoming document belongs. For a new document type, the user must provide a description of the new document type, in order to create the new conceptual structure and build links connecting the new document type within the semantic network. The semantic network has two layers, namely class layer and instance layer. The class layer contains the relationships between different types of documents, and the instance layer contains the description of the environment within which the real documents are embedded. A conceptual document is generated after recognizing the incoming document type (class), and it is then passed to document filing. The conceptual documents are organized based on their classes in the document organization and filing. Original documents are stored in a document base. Classes and conceptual documents are stored in the model base as nodes of the semantic network. The newly generated conceptual document is inserted into the semantic network based on its document class.

Users are allowed to browse through the semantic network at either the class level or the instance level. A query language KQL is provided to specify the search criteria for the document class (type), conceptual structure, content and environment including role and domain dependencies. The procedural knowledge concerning the relationships among the documents and the related office components -- as well as the operational properties -- is described in the document retrieval module. The procedural knowledge is formalized by means of a network-based model which shows how documents are created and accessed, how events trigger the procedures, and who executes the procedures.

The domain knowledge for a specific application is represented in terms of rules general rules and navigation rules. The general rules concern the role of documents both in the procedural context of office and in the context of laws and regulations of the application domain. The navigation rules are used by the system for traversing the semantic network during the search process.

TEXPROS has two major advantages over KABIRIA. The first advantage for the TEXPROS is its document modeling. Combining the document type hierarchy, TEXPROS allows users to create their own folder organization to model the document organization in the same way as that in their real office environment. This gives tremendous flexibility to users who may use the system for organizing, storing and managing their own documents practically and conveniently, regardless of their document types. However, KABIRIA provides the document type hierarchy only and forces the documents to be organized according to their document types only. This model is impractical to mimic the users' document filing systems with effective document retrieval in the real world. The second advantage that TEXPROS has over KABIRIA is in its knowledge management. TEXPROS has a knowledge base, including an object base and a domain knowledge base, and a knowledge acquisition and learning mechanism. This knowledge base accumulates knowledge automatically and dynamically (the knowledge is kept up-to-date), which makes TEXPROS a domain independent system and can be used in most kinds of office environments. However, although KABIRIA has procedural knowledge and domain knowledge, its knowledge is managed in a static way, which makes KABIRIA a domain dependent system and difficult to adapt to the changing office environments.

KABIRIA provides a KQL query language to support document retrieval. TEXPROS provides a strong platform to support more powerful query language-based document retrieval, which is the research focus of this dissertation.

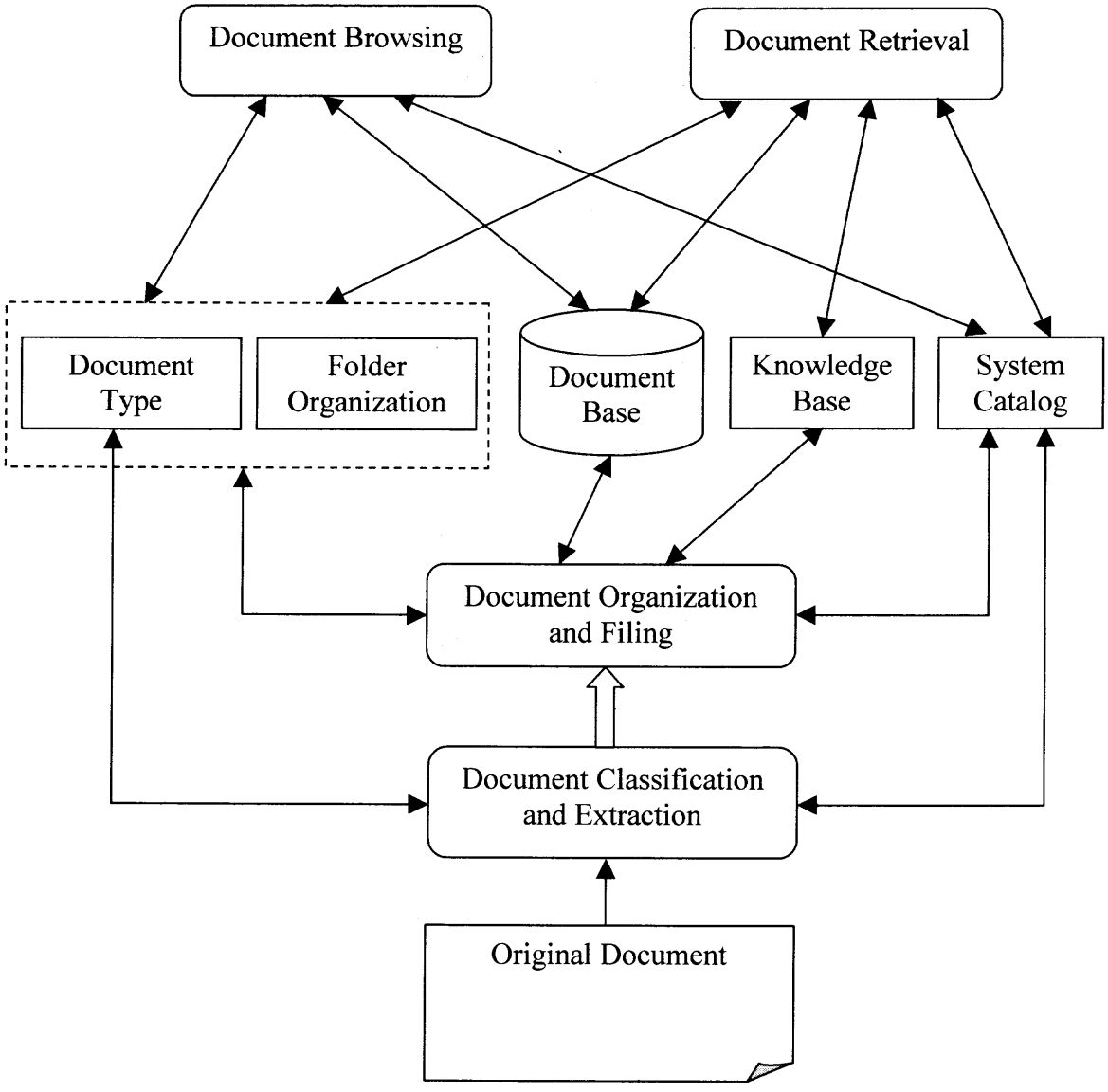


Figure 1.1 TEXPROS architecture at component level

1.4 Document Retrieval vs. Document Browsing

Both document retrieval and document browsing are user interfaces that enable users to navigate and search documents stored in the system.

The document browser supports mainly explorative search. It provides facilities that allow users to navigate the document system and find documents in which they are interested. Normally, the browsing process is an iterative process, and requires full participation from the users. It is usually used when a user has a vague search goal. For users who know exactly what they are interested in, the browser is considerably time-consuming and inefficient.

Document retrieval supports analytical search. It provides users with an interface for specifying their search criteria. A query is then sent to the search engine, which will return the result, if any, to the users. For document retrieval, the query definition and search are separated processes. The users are only involved in defining the query. Upon the arrival of the well-defined query, the system begins to search for a collection of documents that are of interest. Because of this limited participation of the users, the document retrieval process works best for the users when they have a specific search goal.

Document browsing is a process, which is more or less similar to someone going into a library to find something to read. Without knowing exactly what is wanted, he/she starts from the index or search tools, reads the abstract, then goes to the bookshelf and glances over the items of potential interest. This process may be repeated as his/her interest changes, and finally, he/she finds some desired items, which could have nothing to do with his/her initial goal.

Document retrieval is a process that is similar to someone going to a library and asking the librarian for what he/she seeks (by either completing a request form or

answering the librarian's questions). Then the librarian handles the request and brings the items to the user.

Document browsing and document retrieval are complementary and should be interwove, in order to improve the effectiveness and efficiency of a process of retrieving documents.

1.5 Motivation

The users who use TEXPROS to search for documents can be divided into two groups: the one who has vague search goals and the other one who has explicit search goals. As discussed in previous section, the document browser provides an interactive graphic interface that allows users to traverse, perhaps with a vague search goal, the documents stored in the system. For any user group, who has vague search goal, document browser is the most appropriate vehicle to be used at their disposal. One of the disadvantages for using document browser is that it requires the full participation from the users, and therefore, it tends to be less efficient and time consuming for any user group with explicit search goals. The ideal way to support this users group is to allow them specify their requirements using a query language and then the system will return the documents which meet their requirements. A language-based document retrieval is a necessary component for TEXPROS to support its user group who has definitive and explicit search goals. Thus, TEXPROS must have both the document browser and retrieval components in order to meet the users' search needs.

Much research work has been carried out for TEXPROS and a handful of evolutionary solutions have been proposed to make TEXPROS stand out from other peer systems. However, little progress has been made in document retrieval since a simple SQL-like (Select-From-Where) language was proposed in the early stage of TEXPROS [35-37]. Even though, a thesaurus was proposed to support fuzzy keyword searches [53], users are required to know more or less the detailed internal implementation of the

storage base of the system at a professional level. Furthermore, the underlining components such as organization and document classification that support document retrieval were in their early stage of development.

1.5.1 Challenge in Document and Information Retrieval

One of the major challenges in designing and developing a "good" document and information retrieval subsystem is how to deal with the contradiction among its efficiency, accuracy and ease of use. Retrieval using a simple language based query tends to be more efficient and easy to use, but less accurate. Retrieval using a language with more expressive power tends to be more accurate, but less efficient and more difficult to use. Achieving the balance of these three factors becomes one of the major research focuses in designing and developing a "good" document and information retrieval.

1.5.2 Knowledge-Based Retrieval

The research goal of this dissertation is to investigate and develop an approach that supports efficient document retrieval using a more powerful query language. Both efficiency and accuracy of this approach will be investigated. The basic foundation of this approach is the use of the knowledge-based retrieval. It is our claim that, by incorporating a knowledge base into the document retrieval process, the search efficiency and accuracy can be improved tremendously.

Over the last five years, substantial improvements have been made in document organizations and document classifications for TEXPROS. This presents a suitable platform and test bed for studying the knowledge-based approach. The results of our research development will be applied into TEXPROS.

1.6 Scope of Dissertation

The scope of this dissertation is to develop a completely new solution for document retrieval with application to TEXPROS. In Chapter 2 the document retrieval platform including the dual models, three-level document repository and knowledge base is introduced. In Chapter 3, after discussing the knowledge-based technique and document retrieval, the architecture for knowledge-based document retrieval is presented. In Chapter 4 the document query language, which has much more expressive power to support precise query specifications, is proposed. In Chapter 5 the knowledge-based the document processing and search engine is presented. Using the dual model as well as the knowledge collected during document filing, the search engine can efficiently and effectively reduce the search space to a small set of documents, where documents that match the query will be returned to the user. In Chapter 6 the guided search is proposed to help users composing queries. Supported by an intelligent question generator and inference engine, a question base and a predicate-based query composer, the guided search collects the most important information know to a user to retrieve the documents that satisfy user's particular interests. In Chapter 7 conclusions and future work are discussed.

CHAPTER 2

DOCUMENT RETRIEVAL PLATFORM

As shown in Figure. 1.1, taking the dual model concept into consideration, the document retrieval component is built upon the existing document organization and document classification components. This chapter gives a brief introduction to the platform adopted in this dissertation.

2.1 The Dual Model

TEXPROS employs a dual modeling approach for describing, classifying, categorizing, filing and retrieving documents. This document model consists of two hierarchies: a document type hierarchy and a folder organization.

Document type hierarchy depicts the structural organization of the documents. In TEXPROS, documents are partitioned into different classes based on their common properties. Each document class is represented by a frame template, which describes the common properties in terms of attributes of the class and is referred to the document type. The frame templates define how documents should be abstracted and interpreted. To reduce the complexity of models and the redundancy in specifications, frame templates that are related by specialization and generalization are organized as a document type hierarchy.

Based on the document type hierarchy, documents are summarized from the viewpoint of its frame templates into frame instances, which are synopses of the underling documents. With structured format and relatively much smaller sizes, frame instances can be processed much more efficiently than their original documents.

Date: Tue, 19 Sep 2000 15:30:12 -0400 (EDT)
From: Katherine Herbert <kherbert@homer.njit.edu>
To: phd@homer.njit.edu, CSfaculty@homer.njit.edu
Cc: Katherine Herbert <kherbert@homer.njit.edu>,
Connie Perrin <perrin@homer.njit.edu>
Subject: CIS 791 Doctoral Seminar Webboard

Hello!

Recently a webboard was created for CIS 791 Doctoral Seminar. This webboard is intended as a facility to help the PhD students throughout their careers at NJIT. Hopefully, this webboard will become a place where both the PhD students and faculty can meet and discuss their interests as well as contain information pertinent to the completion of the CS Ph.D. degree.

If you are interested in being a part of this webboard, please sign on to it. It is located at http://webboard.njit.edu under the Additional Boards section.

Kathy Herbert

Figure 2.1 (a) An original email

Type	
From	
To	
CC	
Subject	
Date	
Time	

Figure 2.1 (b) Frame template for the email type

Type	Email
From	Katherine Herbert
To	Phd, CSfaculty
CC	Katherine Herbert, Connie Perrin
Subject	CIS 791 Doctoral Seminar Webboard
Date	September 19, 2000
Time	15:30:12

Figure 2.1 (c) Frame instance of the email

In addition to a document type hierarchy, a folder organization is used to describe how documents are managed and organized in a real-world office environment. The folder organization is defined by a user corresponding to his/her view of the document organization, which is obtained by repeatedly dividing documents for particular areas of discourse into groups. Each folder has a user-defined criterion to govern the automatic document filing. The folder organization provides efficient frame instance access by limiting the search to those frame instances of a specific document type in a folder that appears to be most relevant to queries in a collection of frame instances. Figure 2.1 shows the frame template and frame instance of a viewgraph of an email.

Both the document type hierarchy and the folder organization are flexible and user-oriented. With the user-oriented dual model, TEXPROS can interpret and organize documents in the same way as users expected. This encourages users to provide more useful information during retrieval, which will in turn improve the efficiency and effectiveness.

2.2 Three-Level Document Repository

In this dissertation, a three-level architecture of a document repository [15, 17] is chosen from previous proposed solutions to organize and store documents, which is depicted in Figure 2.2.

The first level storage contains original documents, which are physically stored on disks or any other storage media.

The second level storage contains frame instances, which are physically stored in units of bookcases. Bookcases are organized based on the document type hierarchy. Each frame template (i.e. a document type) can have a corresponding bookcase. Each bookcase may contain multiple boxes. Each box contains all frame instances that satisfy certain predicates. Analogous to the inverted indexing, each frame instance has a pointer to its

corresponding original document, and, also, it contains the most relevant information of the document that is pertinent to the user.

The third level is the folder organization. Each folder is a virtual repository for a set of frame instances that only store pointers to the frame instances at the second level.

Since the folder organization can be created using user-defined predicates, it is flexible and user-oriented. The links between each level can be built automatically in predicate-driven filing process. Also, in the process of creating the folder organization and the three-level repository, a knowledge base is created containing the user's knowledge of how the folders are organized as well as the objects involved in the domain organization.

The advantages of adopting this three-level document repository architecture as the document organization for document retrieval can be summarized as follows:

- It supports document retrieval based on both frame instance and frame template.
- It supports fast and efficient document retrieval by incorporating the dual models.
- It allows information about elements and their associations among them from various levels of storage to be stored in the knowledge base to improve document retrieval effectiveness.
- It supports predicate-based query language.

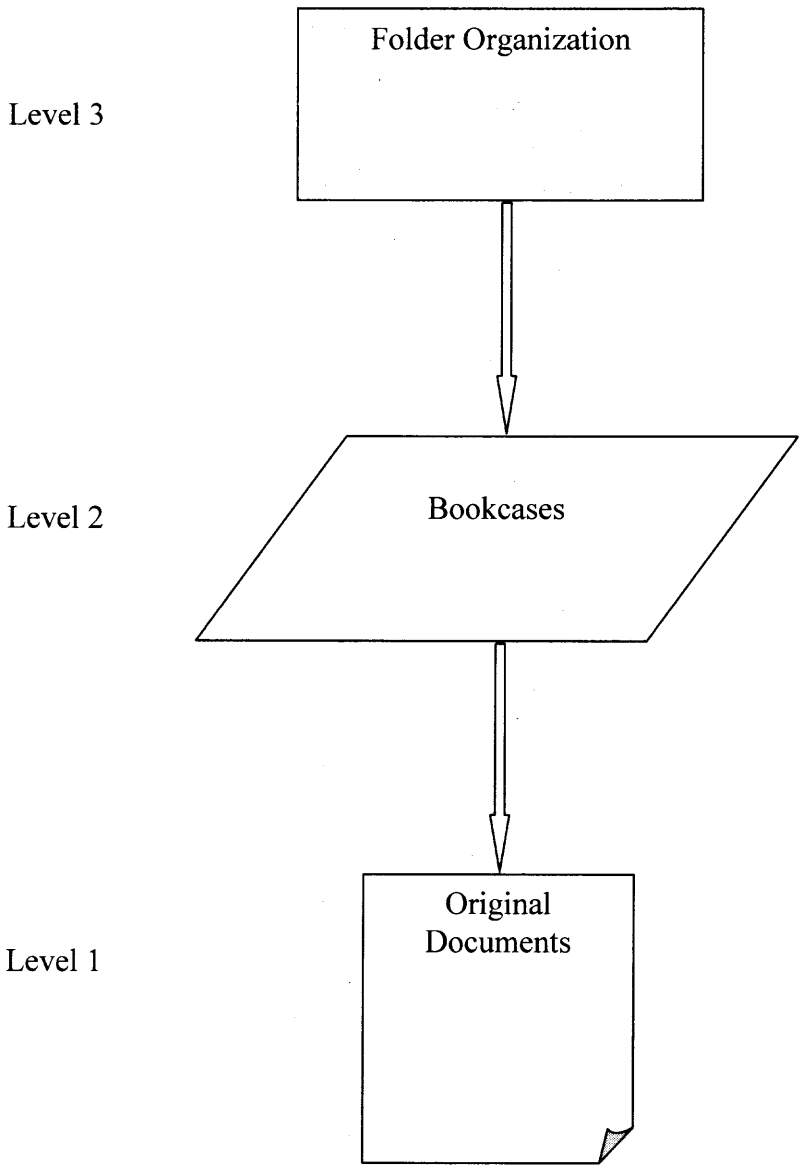


Figure 2.2 Three-level document repository

2.3 Knowledge Base

The knowledge base contains the knowledge of how to automatically organize and file documents into the folder organization created by the user. The document filing is a self-learned and automated process. Knowledge gathered and learned during the document filing process is stored in the knowledge base.

The knowledge base plays a major role for filing documents in three ways. Firstly, the information that is useful to find the documents is provided. This helps a user write precise and efficient queries. Secondly, Folders that contain the user-wanted documents can be searched and located. Locating the relevant folders, which possibly contain the needed documents reduces the search space. Therefore, the knowledge base could enhance the search efficiency. Finally, the knowledge base supports an automated process for examining which documents satisfy the search criteria, and thus to prevent irrelevant documents being returned, which in turn improves the effectiveness.

The knowledge base consists of a domain knowledge base and an object base. The domain knowledge base contains the knowledge of the application domain. A domain may consist of subdomains, and, in turn, a subdomain may have subdomains, and so on. The subdomain knowledge is a collection of well-structured information of the subdomains that an application domain has and of their relationships among these subdomains is stored in the domain organization. In addition, a collection of well-structured information of the properties of the subdomains and their relationships is stored in the domain knowledge base as the property relations. Since the frame instances are organized as a user-defined folder organization, the folder organization can be varied from users to users. Different folder organizations have their own domain knowledge. However, the domain knowledge for the domain knowledge base of a user-defined folder organization can be extracted from the folder organization.

The object base contains a collection of well-structured information or facts about the objects that appear in the user-defined predicates of their folder organization. The object base is domain dependent. In different application domains, the object base deals with different objects, and therefore, contains different knowledge. The object base consists of a set of object pages. An object page is one-to-one associated with an object. The object page for an object contains a collection of well-structured facts about the object in terms of attributes (property names) and their values. Building an object base is controlled by a learning agent using predefined learning topics and learning rules.

2.4 Knowledge-Based Evaluation Engine

To automate document filing, a knowledge-based predicate evaluation engine is implemented to determine whether a frame instance satisfies a given predicate. The engine is composed of a control module, two evaluation modules, an object base, a domain knowledge base and an inference engine. The architecture of the knowledge-based evaluation engine for filing can be depicted as in Figure. 2.3. First of all, the incoming user-defined predicate is parsed and broken into predicate clauses and constraints by the control module. This module also controls the other modules and makes the final decision. The process of evaluating predicate clauses can be divided into three stages. At the very first stage, the evaluation engine examines the first level predicate clauses using information contained in the system catalog. At the second stage, the evaluation engine II accesses the properties of objects from the object base and information regarding the folder organization and its contents from the system catalog to examine the second level predicate clauses. Any second level predicate clauses will be passed to the third stage, whenever the evaluation engine II failed to evaluate them. At the third stage, the inference engine carries out the further evaluation using any facts that are related to their (i.e., predicate clauses) application domain and the properties of the objects within their application domain.

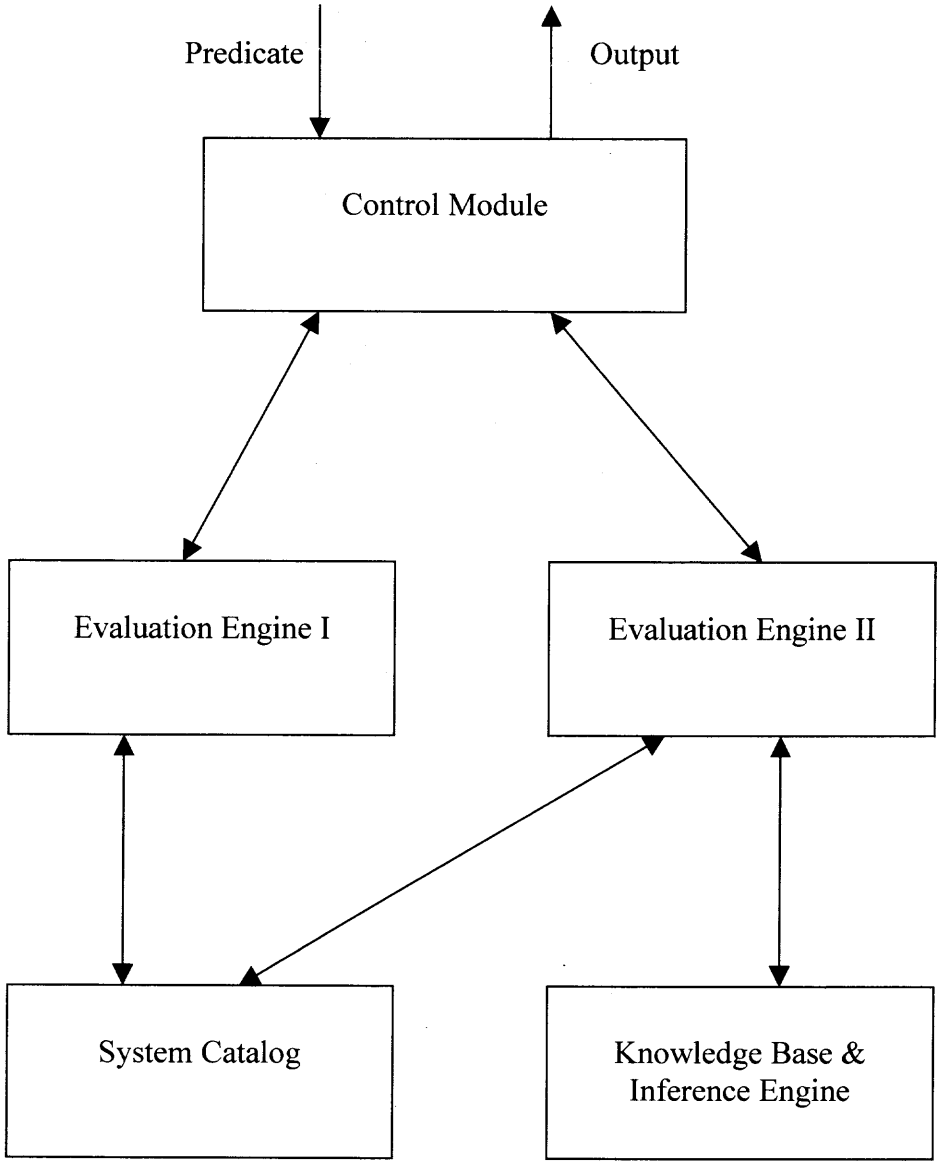


Figure 2.3 Knowledge-based predicate evaluation engine for filing

CHAPTER 3

KNOWLEDGE-BASED DOCUMENT RETRIEVAL

In this chapter, the usage of artificial intelligent and knowledge-based techniques in information and document retrieval is discussed. A knowledge-based document retrieval architecture is presented to provide efficient and effective document retrieval for TEXPROS.

3.1 Information and Document Retrieval

Indexing is a widely used technique in text-based search. The major problem of an indexed search is the ineffectiveness. As the document base grows, the indexed search tends to return more documents. It is very common that a typical keyword search over the Internet identifies hundreds or even thousands of documents; some of these documents are of interest to the user, but most of the these documents are of no interest to the user at all. This sometimes makes a search ineffective, and even worse, the results obtained from the search could be inapplicable, because it may not be practical or acceptable to users to examine the returned documents one by one.

A solution for dealing with such a chaotic phenomenon is ranking the returned documents according to their degree of significance or other factors. Many search tools can be used for ranking the return documents based on pre-defined criteria. However, to achieve an acceptable ranking, the search engine has to understand what the user is looking for. Understanding what the user is looking for can be achieved, if the system could provide users with a powerful query language that allows them to specify a more precise query using feedback mechanism between the user and their system. The keyword search is obviously insufficient. It is difficult for keyword search to verify whether a document is related to a given keyword that does not appear in it. As a matter

of fact keywords are not the only information that users want to use for retrieving documents.

However, the problem of efficiency of document retrieval, and of ease of use could be arisen if the system adopts a powerful query language for users to define any complex and precise queries. With knowledge of what a user is looking for, the search engine also needs to analyze the returned documents in order to make an appropriate ranking, which will decrease the efficiency when the result set is large. Also, composing a complex and precise query using more powerful language may not be an easy job for the average users.

A challenge in the information and document retrieval field is to achieve an appropriate balance among the conflicting priorities of the search efficiency, the effectiveness of retrieval, and the ease of use. A search using a simple query language -- with less expressive power -- tends to be more efficient and easy to use, but less accurate, whereas a search using a complex language -- with more expressive power -- tends to be more accurate and precise, but less efficient and more difficult to use. Finding a methodology or an approach that provides the balance among the search efficiency and effectiveness as well as the ease of use in order to give reasonably good solutions is the major focus of this research.

One of the directions in dealing with this challenge is to adopt a sophisticated language with reasonable expression power for ensuring the accuracy of document retrieval, and apply artificial intelligent techniques, especially the knowledge-based technique, in the information and document retrieval for improving the search efficiency.

3.2 Knowledge-Based Technique

With the advances of artificial intelligence, knowledge-based techniques began to play a critical role in many research fields, including the field of information and document

retrieval. Knowledge-based techniques are used to support query processing, text and image understanding, classification, categorization, etc.

In B. P. McCune, R. M. Tong, J. S. Dean and D. G. Shapiro [40], a research prototype software system RUBRIC for conceptual information retrieval was developed. The goal of the system is to provide more automated and relevant access to unformatted textual databases. The approach is to use production rules from artificial intelligence to define a hierarchy of retrieval subtopics, with fuzzy content expressions and specific word phrases at the bottom. RUBRIC allows the definition of detailed queries starting at a conceptual level, partial matching of a query and a document -- selecting only the highest ranked documents -- for presentation to the user along with a detailed explanation of how and why a particular document was selected. Initial experiments indicate that a RUBRIC rule provides better matches to retrievals performed by human judgment than a standard Boolean keyword expression, given equal amounts of effort in defining each.

In H. Chen [7], a research framework on the knowledge-based document retrieval systems was presented. It adopted a semantic network structure to represent subject knowledge and classification scheme knowledge. It also modeled experts' search strategies and user modeling capability as procedural knowledge. A prototype knowledge-based retrieval system METECAT was implemented based on the blackboard architecture. It was able to create a user profile, identify task requirements, suggest heuristics-based search strategies, perform semantic-based search assistance, and assist online query refinement.

KABIRIA[4] takes into account the knowledge of a documents' environment, including the roles of a document within an office and its dependence on law, regulations, and habits of the application domain. A document retrieval model is proposed that is based on the representation of knowledge describing the semantic contents of documents, the way in which documents are managed by their procedures and by people in the office, and the application domain where the office operates. The domain knowledge is

represented through rules including general rules and navigation rules, in order to relate the components of the conceptual document model and the document retrieval model to the rules and regulations that are held within a specific domain. The general rules are related to the role of documents both in the procedural context of office and in the context of laws and regulations of the application domain. The navigation rules are used by the system to traverse the semantic network during the search process. A query language KQL is proposed to specify the search criteria about the document class (type), conceptual structure, content and application domain knowledge including the roles and domain dependencies in a specific application domain.

In S. Chen and J. Wang [9], a knowledge-based approach dealing fuzzy information retrieval is proposed, where interval queries and weighted-interval queries are allowed for document retrieval. The knowledge is represented in a concept matrix. The elements in a concept matrix represent relevant values between concepts. The implicit relevant values between concepts are inferred by the transitive closure of the concept matrix based on fuzzy logic.

In W. W. Chu, C. C. Hsu, A. F. Cárdenas, and R. K. Taira [10], a knowledge-based approach that retrieves medical images by feature and content with spatial and temporal constructs is developed. Knowledge about an image's features is expressed as a hierarchical structure called a Type Abstraction Hierarchy (TAH). The high-level nodes in the TAH represent more general concepts than low-level nodes. Thus, traversing along TAH nodes allows approximate matching by feature and contents if an exact match is not available. In addition to TAH, a knowledge-based semantic image model is proposed that consists of four layers (namely, the raw data layer, the feature and contents layer, the schema layer and the knowledge layer) to represent the various aspects of an image objects' characteristics. A knowledge-based spatial temporal query language (KSTL) is developed that extends ODMG's OQL and supports approximate matching of feature and content, conceptual terms, and temporal logic predicates.

3.3 Knowledge-Based Document Retrieval

TEXPROS provides a suitable platform and test bed for the research on knowledge-based approaches to document retrieval. With the dual models and three-level storage architecture, TEXPROS can support more powerful query language without necessarily reducing efficiency. Using the knowledge collected during document filing, a query preprocessing can be conducted to reduce the search space and focus the document search on a small set of documents. Combined with classical text retrieval methods -- including indexing -- knowledge-based document retrieval can improve both efficiency and effectiveness tremendously. Also, with the support of the dual models, the three-level repository, the knowledge base, the system catalog and the predicate evaluation engine, a user-friendly intelligent interface can be built to make the system easy to use. This section introduces a knowledge-based document retrieval architecture that provides efficient and effective document retrieval as well as ease to use user interface.

3.3.1 Knowledge-Based Document Retrieval Architecture

As depicted in Figure. 3.1, the knowledge-based document retrieval architecture is based on the document retrieval platform described in Chapter 2.

In this architecture, a predicate-based query language is adopted for specifying search criteria. The detailed description of this predicate-based document query language will be given in Chapter 4. Since predicates have much more expressive power, users can specify the search criteria and knowledge about the documents to be retrieved more precisely and accurately. Since document filing in TEXPROS uses predicate-based language for specifying filing criteria, a predicate-based query language allows the document search engine to take advantage of the folder organization and the knowledge collected during filing.

This architecture provides two kinds of user interface for naive users and experienced users. Users can also specify queries using the proposed document retrieval language and then submit them to the query processing and the search engine. For any novice users, the guided search intelligent user interface provides a quick and easy starting point. Instead of requiring users to compose queries, the guided search interface collects information from users through a set of simple questions. The system will automatically generate a predicate-based query after a user answers questions fully. The user can then modify the query. It allows experienced users to use the guided search for composing queries and then to modify them manually.

The guided search component includes a question base, a rule base, a question generator, an inference engine, a predicate-based query composer, and the guided search user interface. They work in a cooperated manner. The guided search begins the process by asking simple questions about any document(s) a user wishes to locate. The question base, generated dynamically based on knowledge of both the user-defined folder organization and the knowledge base, contains all the questions whose answers may help to improve the efficiency. The rule base contains the rules for governing the conversation between the user and the interface. The rules determine which question, if the user knows the answer, is the most important to speed up the retrieval in a particular scenario. The rules also determine when the conversation should be ended if enough information has been collected for improving the efficiency. The intelligent question generator and inference engine, guided by the rules provided by the rule base, dynamically generates subsequent questions from the question base depending on the answers given by the user with respect to previous questions. The predicate-based query composer is used to generate a query using predicate-based query language based on the information collected from the user. The query can then be displayed to the user for evaluation and modification.

The knowledge-based query processing and search engine includes a query parser and an optimizer as well as a knowledge-based document search engine. The query parser and optimizer are used to validate each element in the query, to normalize the query into disjunctive normal form, and to sort the predicates. The optimization process guarantees that the search engine processes the most important predicates in such a way for speeding up the document retrieval. The knowledge-based search engine executes the query and returns those documents that satisfy the search criteria.

The knowledge base, the system catalog, the dual-models and the three-level repository, which are described in Chapter 2, provide support for the guided search component and the search engine component.

3.3.2 Knowledge-Based Document Retrieval Workflow

The workflow of the knowledge-based document retrieval is shown in Figure 3.2. At the input stage, a query using the predicate-based document query language must be composed in order to identify to the system the desired documents. The experienced user can compose the query using the query language directly. The novice user can use the intelligent search tool to compose the query by answering simple questions. Then, the query is submitted to the query parser and optimizer. The knowledge-based search engine takes the optimized query and conducts the search. Finally, the documents that match the user's search requirements are returned to the user.

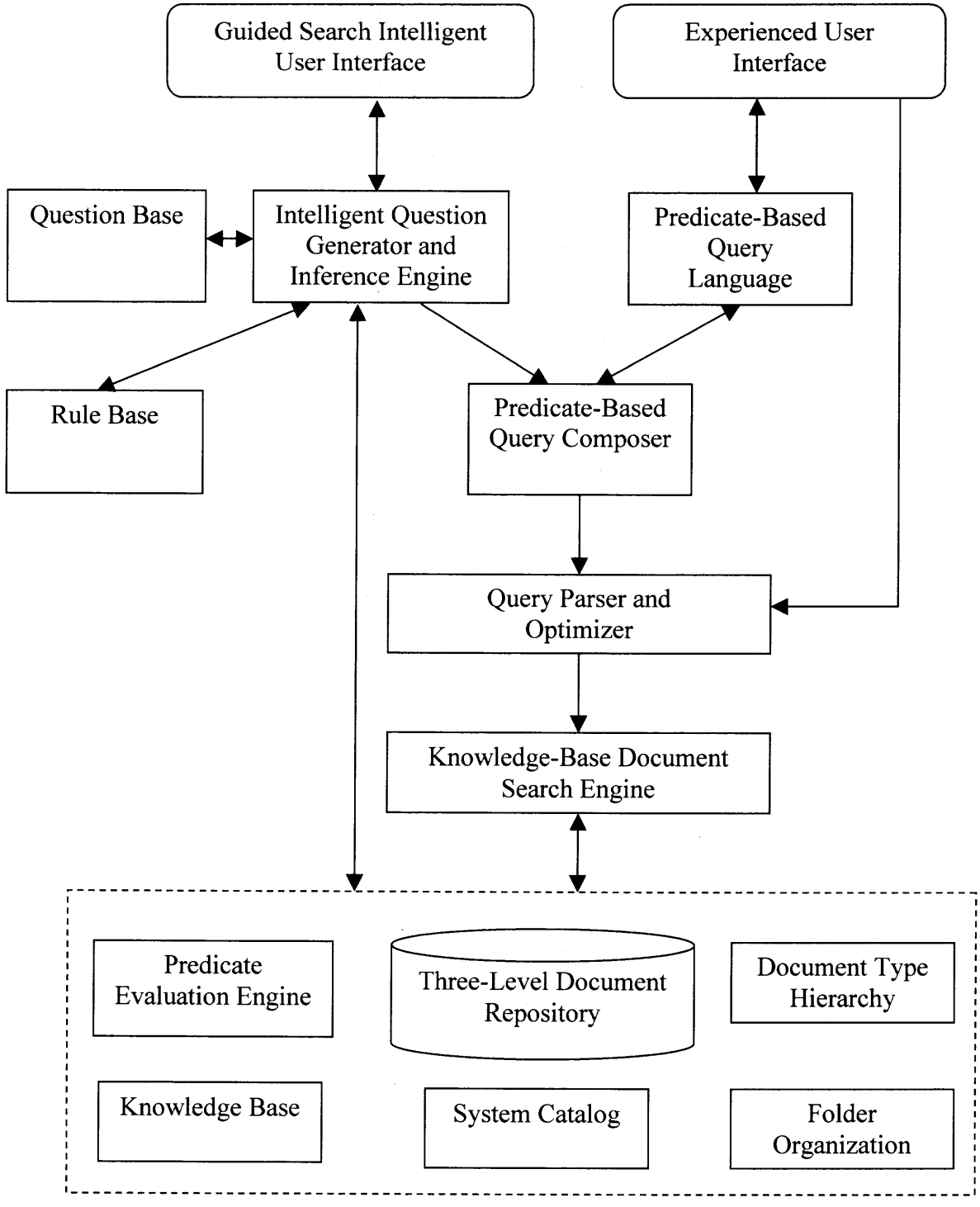


Figure 3.1 Architecture of Knowledge-Based Document Retrieval

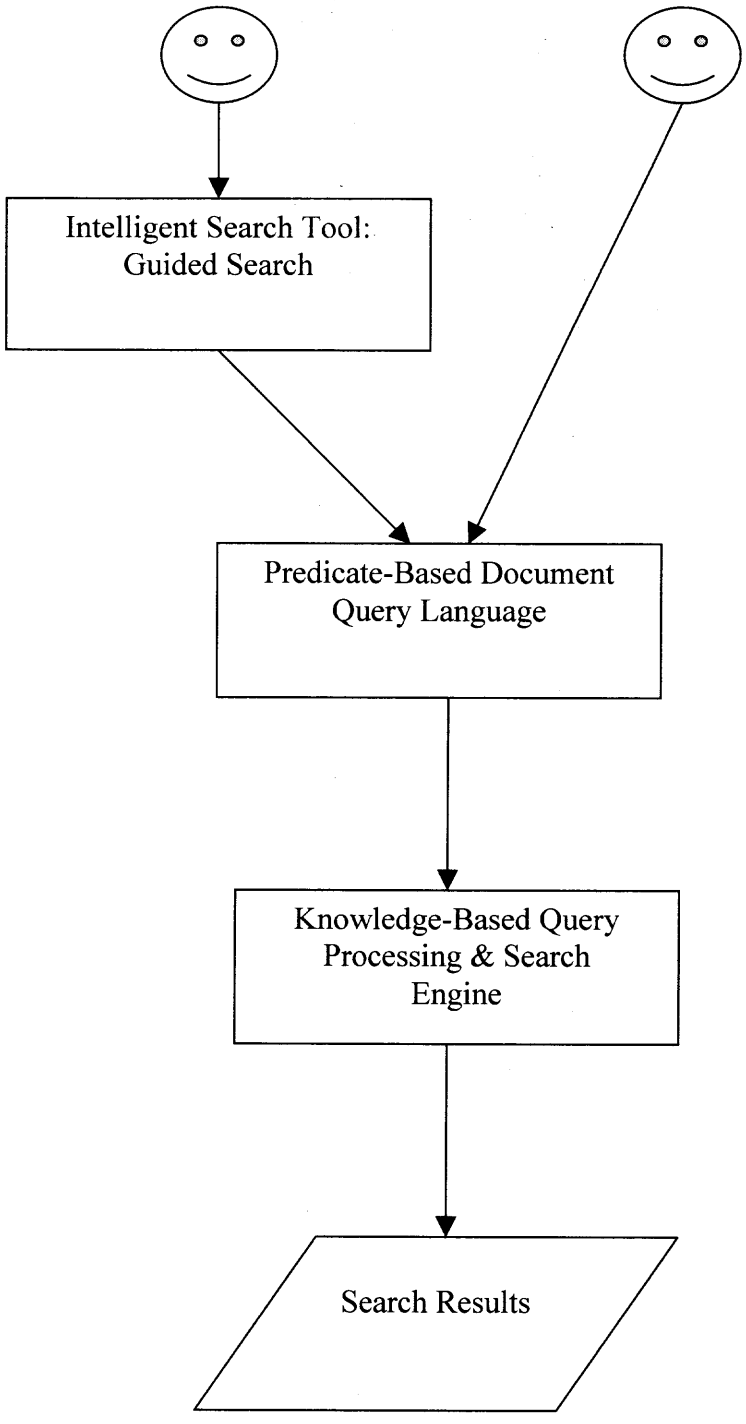


Figure 3.2 Workflow of knowledge-based document retrieval

CHAPTER 4

DOCUMENT QUERY LANGUAGE

One of the major objectives of this dissertation is to investigate and develop a knowledge-based document retrieval approach with an application to TEXPROS. This approach should be able to balance the three conflicting factors of efficiency, effectiveness and ease of use to provide a good solution in document retrieval. In order to allow the user to communicate with the system and tell the system his/her specific search goal, a language is necessary to support the interactive and precise communications between the user and the system. The user should be able to use the language, in a natural and concise way, to specify the search criteria and the knowledge about the documents. In this chapter, the predicate-based document query language is presented to support accurate query specification for knowledge-based document retrieval.

4.1 Query Language Requirements

The query language is used to formalize users' search request in such a way that the search engine can understand and process it. Generally speaking, the query language must have enough expressive power to specify any search requests that the search engine supports. In the mean time, the query language should be easy to use and understand. More specifically in TEXPROS, the query language should be able to describe a document to the extent that the document model can support. This allows users to search a document based on any content that has been captured by the document model. In other words, the query language should be powerful to express and to describe the dual models, namely the document type hierarchy and the folder organization, which are defined by the users.

In addition, the query language should be able to specify any knowledge of document that is used in document filing. The folder organization organizes documents into folders according to user-specified filing criteria, which helps to improve the efficiency by allowing document search through a particular folder. Taking full advantage of the folder organization, the query language should be powerful enough to express and describe the user-defined folder organization including the document filing criteria.

4.2 Keyword Search vs. Predicate-Based Query Language

Keyword search is the most common and easiest way to retrieve particular information from data sources. Since a keyword search has very limited expressive power for describing the search criteria and the search process is simply a string matching, its results normally contain "noisy" information, namely most of the information is irrelevant to users' interests. For example, a user is looking for a letter sent by a professor. The user may get a lot of irrelevant documents, where the words "letter" and "professor" appear as part of the content of each of the documents, if the user issues the query using words "letter" and "professor" as the search keyword. This is because the term "letter" and "processor" are conceptual information, not the content of the letter. Keyword search is obviously insufficient to support sophisticate and precise document search.

Comparing with the keywords, a predicate has much more expressive power to describe the search criteria and users' knowledge about the documents precisely and accurately. Upon receiving predicates specified by the predicate-based language, the search can return results that satisfy users' particular interests to a greatest extent of satisfaction. But the search process requires a more complicated platform to support its use.

The TEXPROS platform consists of a three-level document repository, a predicate-driven folder organization, a knowledge base, and the dual-model. The predicate-based query language is expressive-powerful enough to describe this platform.

4.3 Predicate-Based Query Language

In this dissertation, a predicate-based language is adopted as the query language. The main reason is that while predicate-based language has reasonable expressive power, it is easy to use and understand due to its simple syntax. Another reason is that a predicate can be easily translated into terms specified by the natural language. It is possible to develop a more friendly user interface for inexperienced users and this interface is called the guided search, which will be introduced in Chapter 6. In this section, the specification of the predicate-based query language for knowledge-based document retrieval is presented.

A predicate-based language [15, 17] is used in document filing in TEXPROS for specifying filing criteria. To keep all information consistent within the system, and, more importantly, to take full advantages of the knowledge base and folder organization that are created during document filing process, the predicate-based query language is defined based on predicate-based filing language. Modifications and extensions are made to make the predicate-based query language suitable for document retrieval.

Predicates are statements about objects. In this query language, two kinds of objects are allowed to appear in predicates. One is the frame instance. The other is the object that is relevant to the frame instance.

Definition 4.1 (Pattern) A pattern is a format that defines a specific way to convert a string to another. The following symbols can be used for specifying a pattern:

? : one character

* : any number of characters

: one character that can be ignored

- : any number of characters that can be ignored

Definition 4.2 (First Level Predicate Clause) A first level predicate clause has the format p(FI, v[, r]) where

1. p is the name of the predicate clause and can be an attribute of the frame instance FI;
2. FI is a frame instance, which is a reserved keyword by the system;
3. v is either a value or a variable;
4. If v is a variable, then r can be given as a pattern.

The first level predicate clauses are used to describe the characteristics of the frame instance FI, which is defined based on its associated frame template stored in the document type hierarchy. Each first level predicate clause represents a single attribute/value pair. Therefore, a frame instance can be represented by a set of first level predicate clauses. For example, Type (FI, Letter) is to denote the attribute Type of the frame instance FI having the value Letter.

This definition is derived from the First Level Predicate Clause definition in document filing language [15]. The difference is that the first parameter in the clause is a reserved keyword, not an arbitrary variable. The purpose of this change is to make it suitable for specifying queries to search for documents.

Definition 4.3 (Second Level Predicate Clause) A second level predicate clause has the format $p(x, v[, r])$ where

1. p is the name of the predicate clause and can be a property name of the object x;
2. x is an object;
3. v is either a value or a variable;
4. If v is a variable, then r can be given as a pattern.

The second level predicate clauses are used to specify the properties of an object that is related to a frame instance. Its definition is based on the structure of the knowledge base. Each second level predicate clause represents a single property/value pair of a given object. Considering the relation between an object and a frame instance, a second level predicate clause also represents a piece of knowledge of a document. For example, Occupation (Sender, Professor) denotes that the Occupation of the object Sender is Professor. If it appears in a query, then the requested documents are sent from a professor.

This definition is the same as the Second Level Predicate Clause definition in document filing language [15]. If the second parameter in a first or second level predicate clause is a value, the first or second level predicate clause is called a goal predicate clause. If the second parameter in a first or second level predicate clause is a variable, the first or second level predicate clause is called an assignment predicate clause. A goal predicate is a true or false statement. An assignment predicate clause is to assign a value to its second parameter, which makes the predicate clause true. The pattern is used to specify particular matching needs. For instance, the first order predicate clause Date (FI, v, ##/##/????) assigns the year of the value of the attribute Date to the variable v by using pattern ##/##/???? as the third parameter.

Definition 4.4 (Predicate Constraint) A predicate constraint is a relation among variables and values using the operators in {"=", "≠", "<", "≤", ">", "≥", "∈", "∉"}

This definition is derived from the Predicate Constraint definition in document filing language [15]. The difference is that the "∉" is added to make it a complete operator set.

Definition 4.5 (Atomic Predicate) An atomic predicate is either a goal predicate or a n-tuple (P1, P2, ..., Pn), where Pi, $1 \leq i \leq n$, is either an assignment predicate clause or a predicate constraint.

This definition is the same as the Atomic Predicate definition in document filing language [15]. (Date(FI, v), v > 01/01/2000) is an example of atomic predicate, which specifies that the attribute date of the frame instance is greater than 01/01/2000.

Definition 4.6 (Logical Operations) A logical operation is a relation among the TRUE, FALSE and atomic predicates using operators in $\{\wedge, \vee, \neg\}$, where $\wedge$ stands for AND operation; $\vee$ stands for OR operation; $\neg$ stands for NOT operation.

Definition 4.7 (Predicate)

1. A truth value TRUE or FALSE is a predicate.
2. An atomic predicate is a predicate.
3. If P is a predicate, then $\neg$ P is a predicate.
4. If P and Q are predicates, the (P $\wedge$ Q) and (P $\vee$ Q) are predicates.

As to the query discussed earlier in this section, it can be expressed with the proposed document query language precisely as

(Type(FI, Letter) $\wedge$ Occupation(Sender, Professor))

4.4 Discussion

The document retrieval language is proposed for users to search for the documents stored in the system. With this document retrieval language, users can specify both searching criteria and their knowledge about the documents to be searched in a precise and accurate manner, which is much better than simple keyword search. In designing this document retrieval language, the balance between the characteristics of the language's simplicity and its expressive power is taken into consideration. The more complex the language is, the more difficult it is to use and the less efficient it will be to process.

However, the language has to be more complex than simple keywords in order to support the effective document retrieval. The goal of designing the document retrieval language is to preserve the language's simplicity without loosing its necessary expressive power for formulating a precise query specification. To keep consistent with the document filing language, this document retrieval language is based on the general First Order Predicate Logic (FOPL). The limitation of FOPL is that the object can not be a predicate. Certain modifications are made to tailor it for document retrieval purpose. In this section, a summary about the major differences between the document retrieval language and FOPL language is given.

One of the major differences is that the restrictions are enforced on the objects involved in the predicate specification. Only the frame instances and the objects that are related to the documents are valid objects. Also, the concepts of first order predicate clause and second order predicate clause are introduced to describe the properties of the frame instances and related objects respectively, which enhance the restriction on the objects. In FOPL, any objects can be in the predicates. The evaluation of the predicate requires redefining the knowledge of the involved the objects. The knowledge base and the dual model, including the document type hierarchy and folder organization in TEXPROS, contain the information about the frame instance and the object that related to the documents only. The objects that are irrelevant to the documents are useless for the document retrieval. So, restricting the objects involved in the predicates can simplify the predicate specification and in turn speed up the query processing. Another major difference is that the proposed document retrieval language has precise syntax including the restrictions on which symbols can be used in specifying predicates. For example the frame instance has to be expressed by using symbol "FI". The precise syntax can unify the query specification and evaluation and in turn to speed up the document retrieval. Also, it guarantees that only the facts that are relevant to document retrieval can be expressed and the predicate can be understood within the application domain.

The third major modification is that only one object is allowed in each predicate clause. The assignment of a predicate clause and the predicate constraints are used to present the relationships among the frame instances and the objects that are related to the frame instances. In FOPL, multiple objects appearing in a single atomic sentence are allowed, which may make it ambiguous. Restricting FOPL to a single object helps the system to interpret the predicate in the same way as users expected and in turn to improve the search efficiency and effectiveness. The fourth major difference is that the concept of attribute and value in the predicate specification is introduced. A predicate clause describes a property of an object in terms of attribute-value pair. This simplifies the predicate evaluation and, in turn, to reduce the occurrence of misinterpretations. The last major difference is that the universal quantifier and existential quantifier are not allowed in the proposed document retrieval language. Instead of the quantifier, variables in the proposed predicate specification are defined by assignment predicate clauses. The universal quantifier is rarely needed. Eliminating the use of quantifiers can simplify the predicate evaluation and speed up the document retrieval process.

CHAPTER 5

KNOWLEDGE-BASED QUERY PROCESSING AND SEARCH ENGINE

As discussed in Chapter 3, with the dual models, three-level storage architecture and knowledge base, TEXPROS can support precise query specifications using the proposed predicate-based document retrieval language without necessarily affecting system efficiency. The solution is a knowledge-based document retrieval approach. Using the knowledge collected during document filing, the search space can be reduced significantly. Combined with classical text-retrieval methods including indexing, knowledge-based document retrieval can improve tremendously both efficiency and effectiveness. In this chapter, algorithms, theoretical proofs, performance analysis and workflow about the knowledge-based query processing and search engine are given.

5.1 Query Parser and Optimizer

As shown in Figure 3.1, there are two ways for users to compose a query that specifies the search criteria and the knowledge about the documents to be retrieved. The experienced user can compose a query using the proposed document query language directly, while the new users can compose the query through the intelligent tool guided search. Upon the arrival of a query specification, the query is passed to the query parser and optimizer, which is used to validate each element in the query and to normalize the query into disjunctive normal form as well as to optimize the query. The workflow is as follows:

1. Scan the query by isolating operators and atomic predicates.
2. Validate all operators.
3. Validate all atomic predicates.
4. Normalize the query into disjunctive normal form.

5. Optimize the query by re-ordering the predicates in each conjunctive element ("AND" clauses) to make the first-level predicates preceding the second-level predicates.

Since first-level predicates are used to specify the document type and other attributes of frame instances, processing first-level predicates before second-level predicates can more rapidly narrow the search space.

5.2 Knowledge-Based Document Search Engine

The knowledge-based document search engine is the core component in the proposed knowledge-based document retrieval architecture. The normalized and optimized query is the input of the knowledge-based document search engine. The output of the search engine is the search results that satisfy the search requirements specified in the query. The knowledge-based document search engine uses all the resources in the platform namely, the document type hierarchy, the folder organization, the three-level document repository, the knowledge base, the predicate evaluation engine and the system catalog in order to provide the best match results to the user.

5.2.1 Search Strategy

Two most important criteria for measuring document retrievals are the efficiency and effectiveness. More research in document retrieval has been focused on the search efficiency. However, as the document base size grows, the search effectiveness becomes more and more critical. It is very common that a typical keyword search over the Internet gets hundreds or even thousands documents returned. This sometimes makes the search useless, because it may not be practically or acceptable for users to examine the returned documents one by one.

The goal of the document retrieval process in this dissertation is to improve both efficiency and effectiveness. One of the reasons for the poor accuracy of document

retrieval is the lack of a powerful query language that allows users to specify more precisely the desired documents. This was addressed in Chapter 4 with a predicate-based query language, which allows users to specify their search criteria from various aspects and levels. Another reason is that documents are returned when they are identified as being related to the query. Although documents are usually ranked according to the degree of query relevancy, users still face a large quantity of returned documents. A direct way to improve the effectiveness is to examine each document against the query, and eliminate any obtained documents that do not satisfy the query before return them to the user. There are two major challenges to doing so. The first is the need for precision in the query so that the search criteria can be effectively evaluated against each document within the search space. In this dissertation, the knowledge base generated during document filing is used as the knowledge base to support document retrieval. The second challenge is the one of efficiency. Examining each document would slow down the search process. To address this problem, a knowledge-based query preprocessing is performed to reduce the search space to a small set of documents. With dual models, generating a much smaller and complete set of relevant documents that match the search criteria is possible.

There are two entries to reduce the search space. One is from the document type hierarchy. Identifying the document type enables the search to be concentrated on documents of the type only. Checking the system catalog with the information specified in the query identifies the document type. Obviously, a deeper node (ideally a leave node) in the document type hierarchy is preferred since it represents a smaller set of documents. While users tend to remember a more general document type, it is important for the search engine to identify the most specific document type based on the information specified in the query. According to the document storage architecture, documents of the same type are stored together. This makes it efficient to locate the documents once their type is identified. With the help of document storage architecture, the search space can be

further reduced to a particular box, a subset of the document type, if the common feature of the box holds according to the search criteria.

The other entry is from the folder organization. In the real world, documents are organized into folders. Each of the folders contains documents related for a specific purpose. Documents can be found quickly by looking at a specific folder, as long as the right folder is identified. The folder organization is the users' perspective of document organization. From users' point of view, documents are organized based on predefined filing criteria. This makes the document search possible simply looking into a specific folder. The key issue here is how to efficiently identify the smallest folder and a subset of an identified folder that contains all the relevant documents. The process requires well-structured information stored in the knowledge base and efficient evaluation process from the predicate evaluation engine. The algorithm will be described in later sections. To speed up the process of finding the smallest folder and a subset of an identified folder that contains all the relevant documents, search criteria can be cached to avoid repeating the processing for the same criteria.

Theoretically, the search space can be further reduced when both the document type and the smallest folder are identified. Note that a folder can contain documents of various types. Therefore, given a specific document type, a smaller subfolder containing all the documents of the given document type can be obtained by applying the intersection operator of the document types and the folders. Suppose T is the set of documents identified by examining the document type hierarchy (could be a whole bookcase or a box), and F be the folder identified when evaluating the folder organization. The search can be done on the smaller of T and F, or on the intersection. Which way is better depends on whether the intersection can be generated efficiently. With the three-level document storage architecture that incorporates the dual models, it is more efficient to search on the intersection of T and F. According to the document storage architecture, frame instances are physically stored on second level based on

document type hierarchy. Folders, as logical storage, contain the pointers to the physical locations of the frame instances on the second level. By looking at the pointers, one can tell what document types and which boxes the frame instances are. Therefore, a single scan on the pointers can generate the intersection of T and F. This is much more efficient than document searching on the smaller of T and F, because examining the pointers is much faster than examining documents.

After the search space is reduced to a small set of documents, documents will be matched against the query. Only the documents that satisfy the search criteria will be returned to users. This match process is needed because the preprocessing only guarantees that the set of document candidates contains all the relevant documents. But not all the documents in the set satisfy the query. The process can be knowledge-based or content-based, depending on the criteria in the query. Keyword criteria require matching the keyword in a specific part (attribute) of the documents. For single word matching, indexing technique could be used to avoid runtime word matching. Because the frame instances are structured text-based synopsis of the original documents, runtime word matching is still needed for the part that is not indexed, or for evaluating exact sentence or group of word criteria. If the query contains high-level criteria, knowledge-based matching is required to evaluate whether a document satisfies the criteria.

For an efficient query, which contains necessary useful information to narrow down the search space, the above strategy makes the search very efficient and effective. A less efficient query, such as the one that contains only keyword criteria, will force the search engine either return all the related documents with poor accuracy, or examine a large set of documents which reduces the efficiency. Therefore, an efficient query should contain not only the criteria that can identify the document wanted, but also information that can help the search engine to quickly limit the search on a small set of documents. Note that users define the folder organization and the document type hierarchy. It is reasonable to assume that users can provide information or clue for determining which

folder should contains the relevant documents, and which document type they are looking for. The issue is how to help users to specify such information. This topic will be discussed in the next Chapter, which presents a guided search to help users to input necessary and useful information about the documents they are looking for.

5.2.2 Algorithms

Algorithms are presented based on the search strategy, which constitute the most critical part of the search engine. They can be divided into two groups. The first group of algorithms included in Algorithms 5.1, 5.2 and 5.3 are used to narrow down the search space to a set of frame instances. The second group of algorithms included in Algorithm 5.4 are used to search the original documents that exactly match the search requirements in the narrowed search space (the set of frame instances) generated by the first group of algorithms. In this section, the detailed description of the algorithms is given.

Algorithms 5.1 Let Q be a normalized and optimized query formula that is in disjunctive normal form Q1 $\vee$ ...$\vee$ Qi $\vee$...$\vee$ Qn, i =1..n

1. Generate a set of attributes FA by scanning the elements in query formula Q that contains first level predicates;
2. Generate a set of document types T that contains all attributes in FA by scanning the leaf level in the document type hierarchy;
3. Return T.

This algorithm is used to derive the document types from the query specified by a user. Knowing the document type of documents to be retrieved can help to reduce the search space quickly based on the three-level document repository. Since the attributes specified in the query may not exactly match the attribute definitions in document type hierarchy, the thesaurus in the system catalog and the well-structured information in the knowledge base can be helpful in step 2 to find the document type.

Algorithm 5.2 Let Q be a normalized and optimized query formula that is in disjunctive normal form Q1 ∨ …∨ Qi ∨…∨ Qn, i =1..n. For the given Qi,

1. Place the root of the folder organization into a stack called ST. Use F to record the smallest folder ever placed in ST. Initialize F with the root.
2. While ST is not empty, do the following:
   2.1 Pop up a folder *f* from ST
   2.2 For each child *c* of *f*, Do
      (a) If *c* is connected to folder *f* with and-link, and has not been visited from all the parent folders connected with and-links, mark *c* as visited from *f*. Go to next child.
      (b) Let P be the local predicate of *c*, check whether Qi→ P holds. If yes, push *c* into ST.
   2.3 Compare F with the folder just pushed into ST. If the folder just pushed into ST is smaller than F, then update F with the smaller one.
3. Return F

The target folder is identified as the smallest folder in the folder organization whose global predicate can be inferred from the query. This algorithm is for finding the target folder. The folder returned by this algorithm contains pointers to the frame instances of the candidate original documents that the user searches for. In step 2.2 (b), the local predicate P and Qi are submitted to the knowledge-based predicate evaluation engine for evaluation, which will be addressed later in this chapter.

Algorithm 5.3 Let Q be a normalized and optimized query formula that is in disjunctive normal form Q1 ∨ …∨ Qi ∨…∨ Qn, i =1..n

1. Identify the set of document types T = {Tj} by calling Algorithms 5.1;
2. For each conjunctive element Qi, do

2.1 Find the smallest folder Fj whose global predicate can be inferred from the conjunctive formula Qi by calling Algorithm 5.2;

2.2 If the number of document types in T greater than a threshold, then set FIji equals to Fj and then go to 2.3. Otherwise, For each document type Tj, Do

(a) Identify the bookcase BCj = (Tj, Pj, Bj) for the document type Tj;

(b) Generate a set of boxes whose common features contained in predicate Pj can be inferred from conjunctive element qi, i.e.

$$Cji = \{BXjk \in Bj: Qi \Rightarrow Pj(BXjk), k = 1..m\}$$

(c) Generate a set of frame instances by selecting the frame instances from the set of boxes that have pointers pointing to it from folder fj, i.e.

$$FIji = Fj \cap (\cup BXjk), k = 1..m$$

2.3 Generate a set of frame instances for conjunctive element Qi,, i.e.

$$FIi = \cup FIji$$

3. Generate a set of frame instances for query Q, i.e.

$$FI = \cup FIi$$

4. Return FI

This is the main algorithm that narrows the search space to a set of frame instances that contains all the candidates satisfying the search requirements specified in the normalized and optimized query. Firstly, the Algorithm 5.1 is called to get the set of document types. For each conjunctive element Qi, Algorithm 5.2 is called to get the smallest folders, and then for each document type a set of frame instances is generated by calculating the intersection between the set of folders and the frame instances in boxes at the bookcase. Finally, a set of frame instances is generated and returned by calculating the union of the frame instances generated for each conjunctive element.

Algorithm 5.4 Let Q be a normalized and optimized query formula that is in disjunctive normal form Q1 ∨ ...∨ Qi ∨...∨ Qn, i =1..n, and let FI to be the set of frame instances returned by Algorithm 5.3, do

1. Initialize the result set DS to empty;
2. For each frame instance Fi in FI, do
    2.1 Check if the Fi satisfy any element Qi in Q;
    2.2 If the answer of 2.1 is YES, then add Fi with the link to original document to the result set DS;
    2.3 If the answer of 2.1 is NO, go to 2
3. Return DS

This algorithm is used to find the original documents that exactly match the search requirements specified in the query based on a narrowed search space i.e. a set of frame instances generated by calling the first group of algorithms including Algorithm 5.3, and then 5.1 and 5.2. In step 2.1, the knowledge-based predicate evaluation engine is called for evaluation.

5.2.3 Knowledge-Based Predicate Evaluation Engine for Document Retrieval

As discussed, the Algorithm 5.2 calls the knowledge-based predicate evaluation engine in step 2.2 with input of two predicates. One is the local predicate for the folder organization. Another is the conjunctive element in the normalized and optimized query. However, the inputs of the knowledge-based predicate evaluation engine developed in [15, 17] are a predicate and a frame instance, because it is designed for document filing purpose. Hence, it cannot be used by Algorithm 5.2 directly.

In order to enable the knowledge-based predicate evaluation engine handling both kinds of inputs, the input preprocessor is added to the existing evaluation engine. The algorithm for the input preprocessor is given as follows.

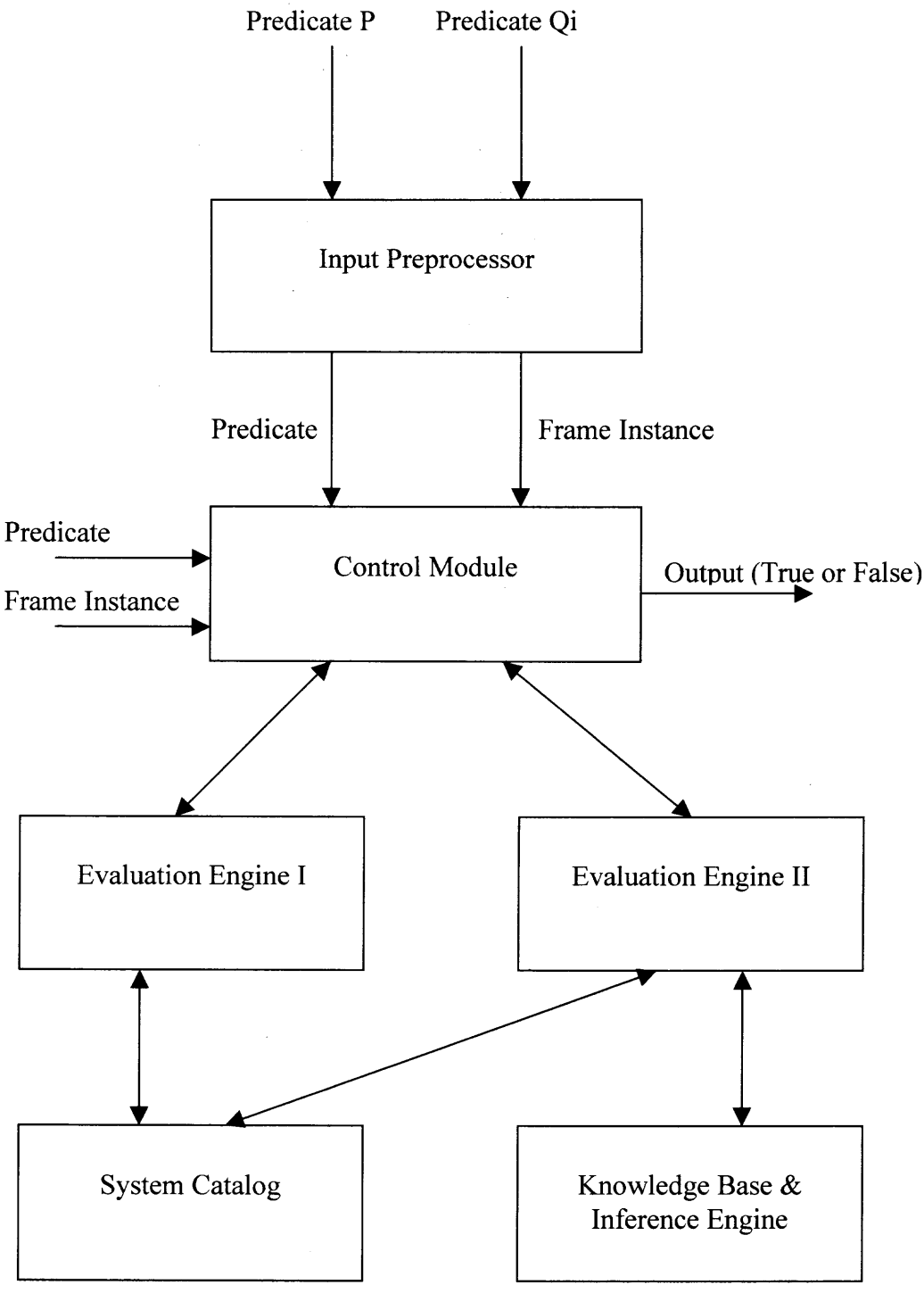


Figure 5.1 Knowledge-based predicate evaluation engine for retrieval

Algorithm 5.5 Let P be the predicate associated with a folder, Let Qi be the disjunctive element Qi in the normalized and optimized query Q, do

1. Generate a set of first level predicates FP by identifying the first level predicate clauses in Qi;
2. If FP is empty, set the frame instance Fj to be NULL, and set the predicate P to be P $\vee \neg$Qi;
3. If FP is not empty, then generate the frame instance Fj by assigning the values contained in FP;
4. Return P and Fj

The architecture of the knowledge-based predicate evaluation engine for document retrieval is shown in Figure 5.1. Two kinds of inputs are supported. One kind of input consists of a predicate and a frame instance. Another kind of input consists of a local predicate of a folder and a conjunctive element.

5.2.4 Query Cache

In order to get the ideal search results, the query refinements happen often after obtaining the search results back from the search engine by submitting the initial query. It is quite typical for an initial query to be too loose or too tight. Hence, the first refinement can be too tight or too loose. This means that the query refinement activities can be repeated several times. Since the initial query and refined queries tend to be similar, buffer the intermediate results can reduce the search time dramatically. Therefore, the concept of query cache is introduced in the search engine to store the intermediate search results of queries that can, in turn, speed up the search for refined or similar queries. The query cache in the search engine consists of a document type cache, a folder cache and a frame instance cache. The details for each of these are given below.

The document type cache is organized in pairs of attribute set and document type set. The element in the attribute set can be any variation of the formal attribute definition in the document hierarchy. The element of the document type set is the formal document type name in the document type hierarchy. The hit in document type cache is checked right after step 1 in Algorithm 5.1. If a hit is found, Algorithm 5.1 returns the document set immediately and skips step 2. If the hit is not found, Algorithm 5.1 moves to step 2. Also, the document type cache is updated right after completing step 2 in Algorithm 5.1.

The folder cache is organized in pairs of a conjunctive formula and a folder. The conjunctive formula is the element in the normalized and optimized query that is converted from the user's specification. The folder is the smallest folder in the folder organization whose global predicate can be inferred from the conjunctive formula. The hit in the folder cache is checked before moving into step 2.1 in Algorithm 5.3. If the hit is found, the Algorithm 5.3 loops into the next conjunctive formula in the query by skipping step 2.1. If the hit is not found, the algorithm 5.3 moves to step 2.1. Also, the frame instance cache is updated right after completing step 2.1 in Algorithm 5.3.

The frame instance cache is organized in pairs of a conjunctive formula along with the document type, and a set of frame instances. The conjunctive formula is the element in the normalized and optimized query that is converted from the user's specification. The element of the set of frame instances is the actual frame instance record in the bookcase. The hit in the folder cache is checked before moving into step 2.2 in Algorithm 5.3. If the hit is found, the Algorithm 5.3 loops into the next document type in the document type set by skipping step 2.2 (a)(b)(c). If the hit is not found, the algorithm 5.3 moves to step 2.2 (a). Also, the frame instance cache is updated right after completing step 2.2 (c) in Algorithm 5.3.

An important issue along with the cache is the maintenance. The size of the cache should be small in order to reduce the overhead by managing the cache. Also, the cache needs to be refreshed periodically by removing the old data, in turn to improve the

performance. When changes happen to the system organization, e. g. changes in the folder organization or in the document type hierarchy or in the knowledge base, the cache must be flushed to guarantee the accuracy.

5.2.5 Theorem and Proof

In the proposed algorithms for the knowledge-based document search engine, Algorithms 5.2 plays critical role in narrowing down the search space to particular folders. A typical concern is does this algorithm overlook any folders containing the references to the documents that satisfy the user's query? In this section, a strict theoretical proof is given to prove that the folder returned by this algorithm is complete with respect to the query.

Definition 5.1 A folder *f* is said to be complete with respect to a query Q if *f* is guaranteed to contain all the documents in the document base that satisfy Q.

Theorem 5.1 Let Q be a query, and P*f* be the global predicate of a folder *f*. Folder *f* is complete with respect to Q if and only if Q→ P*f*.

Proof:

Let *d* be any document in the document base that satisfies Q. If Q → P, then *d* satisfies P. According to the definition of the global predicate, *d* must be in the folder *f*. Therefore, *f* is complete with respect to Q.

If Q → P is not true, then there should be a document *d* such that *d* satisfies Q but does not satisfy P. Therefore, *d* can not be in the folder *f*. According to Definition 5.1, *f* is not complete with respect to Q.

Theorem 5.2 Let Q be a given query, then the folder *f* returned by the Algorithm 5.2 is complete with respect to Q.

Proof:

Assume *f* is not complete to Q. Let *f1, f2, ... fn, f1', f2', ... fm'* be all the parent folders of *f*, where *(fi, f)* (1 <= i <= n) are an "or" links, and *(fi', f)* (1 <= i <= m) are an "and" links. Let P*f*, P*f1*, P*f2*, ..., P*fn*, P*f1'*, P*f2'*, ..., P*fm'* be the global predicates of folder *f, f1, ..., fn, f1', ..., fm'* respectively. Then $P_f = \delta_f \wedge (\sum_{i=1}^{n} P_{fi} \vee \prod_{j=1}^{m} P_{fj'})$, where δ_f is the local predicate of f. According to Theorem 5.1, Q→ P*f* does not hold. Since *f* is in ST, Q → δ_f holds. Therefore, Q→ $(\sum_{i=1}^{n} P_{fi} \vee \prod_{j=1}^{m} P_{fj'})$ does not hold. Hence, there should be a document *d* such that Q is true but $(\sum_{i=1}^{n} P_{fi} \vee \prod_{j=1}^{m} P_{fj'})$ is false. This means that *d* does not satisfy any of *Pf1, ... Pfn,* and all of *Pf1', ... Pfm'*. Therefore, none of *f1, ... fn* is complete to Q. And, at least one of *f1', ... , fm'* is not complete to Q. According to the algorithm, *f* should not have been visited. This violates the facts that *f* is in ST. Hence, *f* is complete to Q.

5.2.6 Performance Analysis

Let k be the number of folders in the folder organization. The complexity of Algorithm 5.2 depends on the number of folders that are visited by the program. In most cases, a leaf folder and the folders along one of the filing path will be visited. In the worst case, all folders may have to be visited. To determine if the local predicate of a folder can be derived from Qi in step 2.2, the predicate evaluation engine will be invoked. Let d be the average time needed by the predicate evaluation engine for evaluating a predicate. According to [15, 17], d is O(1). The complexity of Algorithm 5.2 is O(log k) on average and O(k) in the worst case.

For the Algorithm 5.3, the complexity of step 2.1 is O(log k) on average and O(k) in the worst case. The complexity of step 2.2(a) is O(t) where t is the number of boxes in the bookcase BC. Step 2.2(b) needs O(n) time, where n is the number of frame instances in the folder returned by Algorithm 5.2 . The step 2 has s iterations, where s is the number of conjunction elements in the query Q. So the complexity is O(s (log k + t + n)) on average and O(s (k + t + n)) in the worst case. Assume that s and t are insignificant in compare with the total number of the documents in the system, then the complexity will be O(log k + n) on average and O(k + n) in the worst case.

For Algorithm 5.4, step 2 repeat m times, where m is the number of frame instances in FI. Hence, the complexity of step 2 is O(m).

For the search engine, let C1 be the complexity of Algorithm 5.3, C be the complexity of search engine. In average, m is much smaller than n. So C = C1 + O(m) = O(log k + n). In the worst case, C = C1 + O(m) = O(k+n).

Based on the theoretical complexity analysis above, we can conclude that the proposed knowledge-based search engine is much more effective by supporting more precise and accurate queries, and also very efficient with the complexity to the depth of the folder organization plus the number of frame instances within a folder.

An intelligent interface guided search will be presented in next chapter, in an effort to make the document search easier. Therefore, we can prove that the whole architecture of the proposed knowledge-based document retrieval is a competitive and appealing approach that balances nicely three contradictory factors: the effectiveness, the efficiency and ease of use.

5.2.7 Search Engine Workflow

Figure 5.2 shows the search engine workflow. The input to the search engine is the normalized and optimized query that is converted from the user's original specification. The output of the search engine is a set of the frame instances along with their original

documents. To determine the possible document types of the desired documents, Algorithm 5.2 is called and the document type cache is accessed.

After the set of document types is generated, the workflow moves to the two levels of nested iterations. The outer level of iterations is for each of conjunctive element in the normalized and optimized query. The folder cache is accessed. If no hit in the folder cache, Algorithm 5.2 is called to find the smallest folders whose global predicate can be inferred from the particular conjunctive element. The inner level of iterations is for each document type in the set and the particular conjunctive element that the outer loop iterated on. The frame instance cache is accessed within a single iteration at inner level. After a single iteration at the inner level, a set of frame instances is generate by calculating the intersection between the set of folders and the frame instances in boxes at the bookcase. After completing the inner level of iterations, a set of frame instances that most likely satisfies an element in the normalized and optimized query for all document types in the set is generated by calculating the union of the frame instances generated in each single iteration at the inner level. After completing the outer level of iterations, a set of frame instances that most likely satisfies the normalized and optimized query for all possible document types is generated by calculating the union of the frame instances returned from the inner loops.

Finally, a set of frame instances along with their original documents is returned to the user by examining each element in the candidate frame instance set and adding the frame instances that satisfy the query only into the final search results.

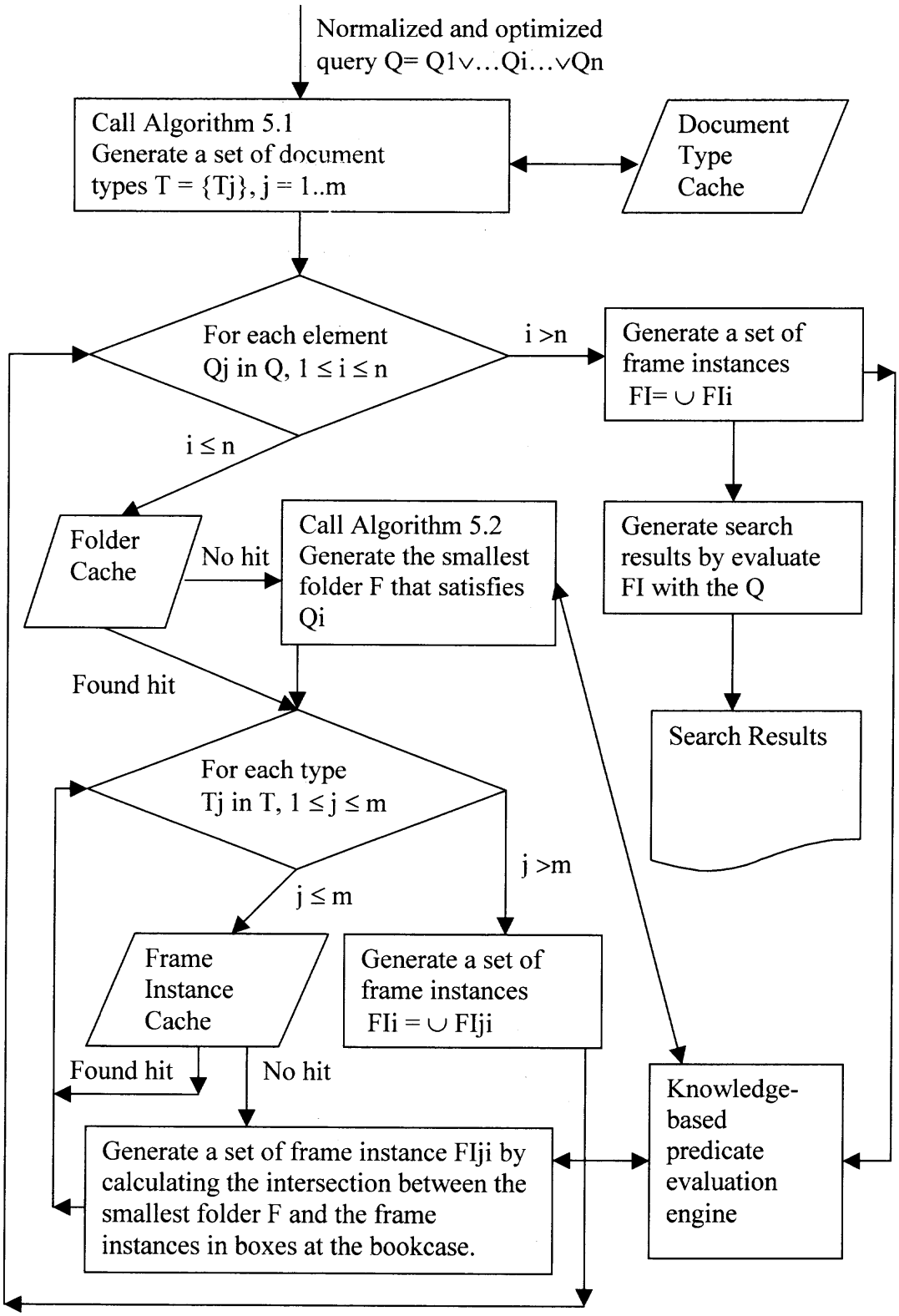


Figure 5.2 Workflow of knowledge-based document search engine

5.3 Example

In this section, an example is given to illustrate the whole process from submitting a user-specified query to returning the search results to the user through the knowledge-based query processing and search engine.

The document type hierarchy for this example is shown in Figure 5.3, which is a typical one for an academic work environment. The folder organization for this example is shown in Figure 5.4, which is a typical one for the NJIT office environment. The boxes in bookcase are organized by date on a monthly basis. The domain knowledge corresponding to the folder organization in Figure 5.4 is shown is Figure 6.1. We further assume that John Smith is a Ph.D. student in CIS department at NJIT, according to the knowledge base.

Search goal:

Find the papers written by John Smith that were published in a conference.

Knowledge about the documents specified by the user:

The co-author of the paper is a faculty member in the CIS department at NJIT.

The subject of the paper is document management.

The conference date is January of 2000.

Knowledge contained in the knowledge base:

John Smith is a Ph.D. student in the CIS department at NJIT

Query in predicate-based query language:

Type (FI, "PAPER") ∧

Author (FI, "JOHN SMITH") ∧

Position (Author, FACULTY.CIS.NJIT) ∧

Subject (FI, "DOCUMENT MANAGEMENT") ∧

ConferenceDate (FI, "01/##/2000")

After composing the query as above, the query is submitted to the query parser and optimizer for query normalization and optimization. The original query is already in normal form. In the optimization, the first order predicates including Subject(FI, "DOCUMENT MANAGEMENT") and ConferenceDate (FI, "01/##/2000") are moved to proceed the second order predicate Position (Author, FACULTY.CIS.NJIT).

Normalized and optimized the query Q:

Type (FI, "PAPER") ∧

Author (FI, "JOHN SMITH") ∧

Subject (FI, "DOCUMENT MANAGEMENT") ∧

ConferenceDate (FI, "01/##/2000") ∧

Position (Author, FACULTY.CIS.NJIT)

The normalized and optimized query Q is passed to the knowledge-based document search engine. Firstly, the search engine narrows down the search space to a set of frame instances that contains all the candidates of the search results. This process follows the steps in Algorithm 5.3 that invokes Algorithm 5.1 to get the document types and invokes Algorithm 5.2 to get the smallest folder. This process is described step by step as below.

Step 1 in Algorithm 5.3:

The Algorithm 5.1 is invoked to generate the set of document types. At the step 1 of Algorithms 5.1, a set of attributes is generated as A = {Type, Author, Subject,

ConferenceDate}. In step 2 of Algorithm 5.1, a set of document types is generated as T = {ProceedingArticles} by scanning the leave of the document type hierarchy in Figure 5. 3, because the "ProceedingArticles" is the only document type that contains all elements in the set of attributes A. Here, the thesaurus is used to find that there is a match between the terms "Paper" and "Article". So, T = {PreceedingArticles} is returned.

Step 2 in Algorithm 5.3:

Since the query Q has only one conjunctive formula and the set of document types has only one document type, the outer loop and the inner loop are executed only once. In the outer loop, the Algorithm 5.2 is invoked to generate the smallest folder that can be inferred from the query Q. In the Algorithm 5.2, the push and pop up operations are carried out repeatedly. The predicate evaluation engine is called to check if the local predicate associated with the child folder can be inferred from query Q. Knowledge base is used to decide which child is pushed into the stack. At the end of Algorithm 5.2, the "Document management" is found to be the smallest folder in the folder organization (Figure 5.4). In the inner loop, the set of frame instances is generated by calculate the intersection of the folder "Document management" and the box with date January 2000.

Step 3 and 4 in Algorithm 5.3:

Since the query Q has only one conjunctive formula, the set of frame instances for the candidates of the results is the intersection of the folder "Document management" and the box with date January 2000.

Final Step:

After generating the set of frame instances that most likely satisfy the query Q, Algorithm 5.4 is called to evaluate each element in the set and returns only the frame instances along with the original documents that make Q true. The knowledge base is used for evaluation.

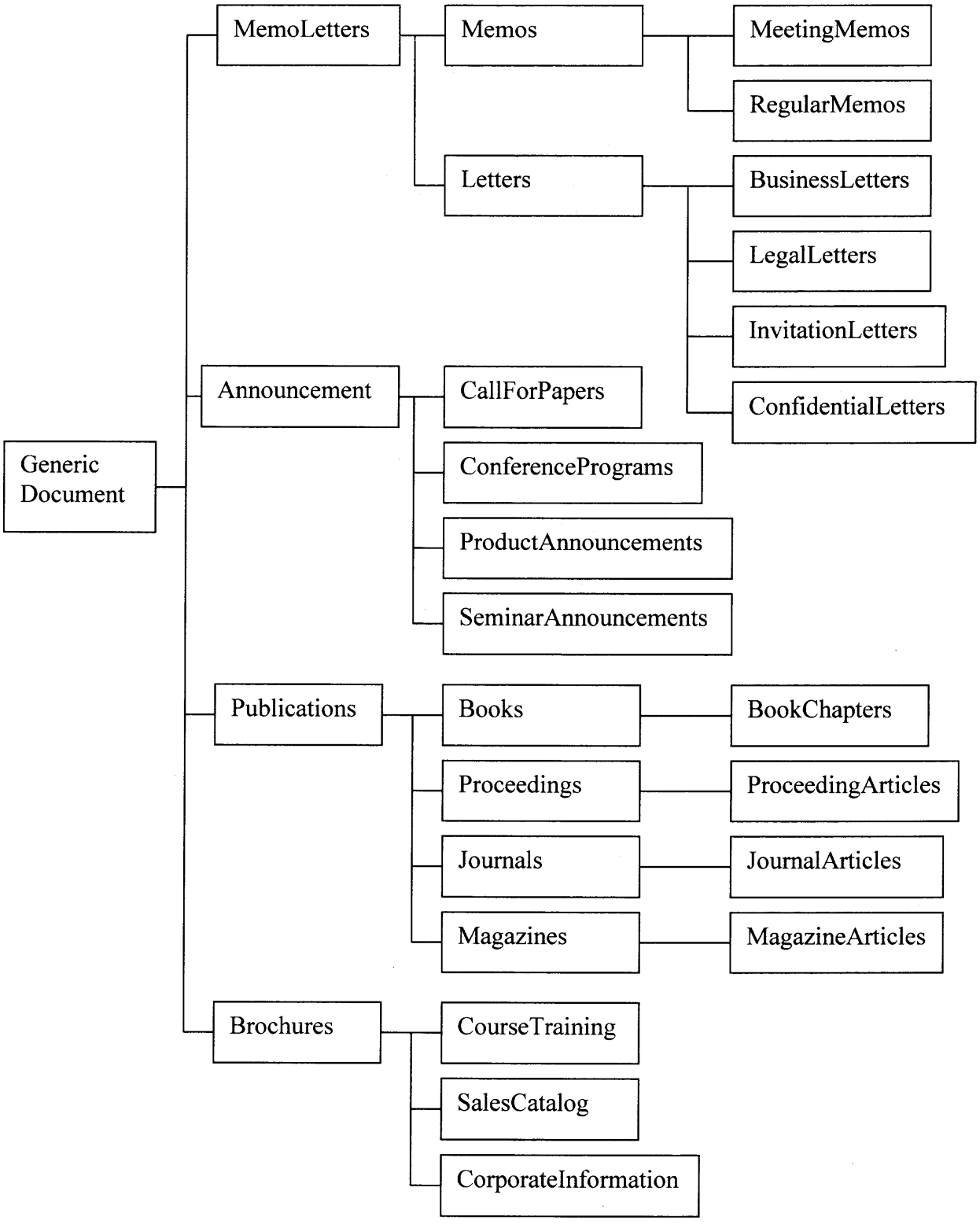


Figure 5.3 A document hierarchy for academic office environment

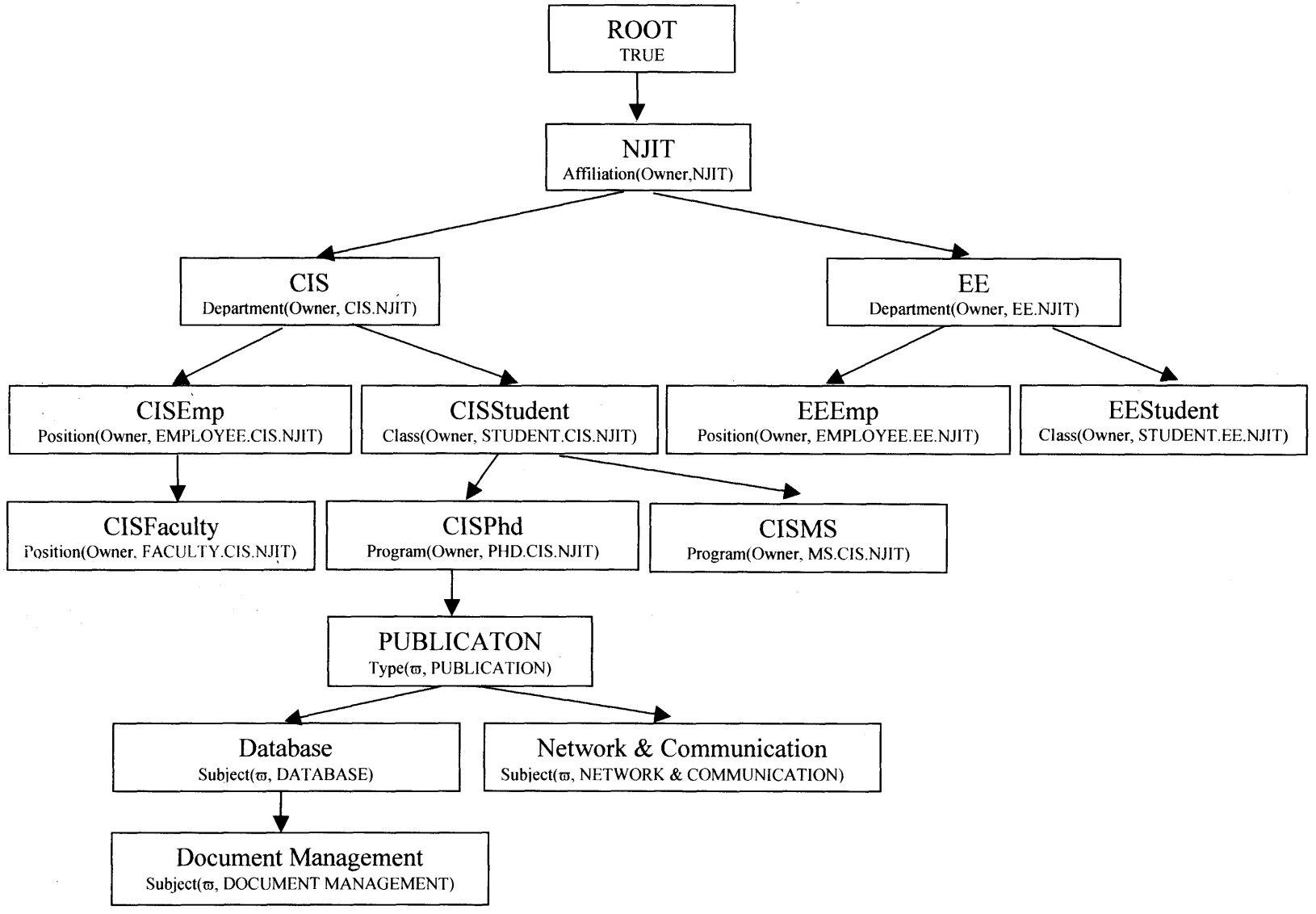


Figure 5.4 A folder organization for a NJIT office environment

CHAPTER 6

INTELLIGENT SEARCH TOOL: GUIDED SEARCH

A guided search is an intelligent user interface for document retrieval. A guided search is used to compose a query through a brief conversation with the user. One of the purposes of a guided search is to help new users who are not familiar with the query language and who know little about the system. The experienced users can also use the guided search as a tool to generate an initial query and then compose the final query based on it. With the assistance of guided search, the experienced users can more easily specify their explicit search goals and minimize the spelling errors.

Another reason for a guided search is to improve the efficiency and effectiveness of document retrieval. In a database, how fast a SQL query can be returned not only depends on database tuning, but also on the query itself. An efficient query can take advantage of the database tuning, and therefore, give a faster return. The efficiency of knowledge-based document retrieval depends on the information collected from the users. Taking advantage of the knowledge base collected during document filing, the guided search can tell users what information is most important for rapidly finding the documents.

A guided search collects information from a user by asking simple questions, with the most important one first. The following questions issued will depend on the user's answer to the previous questions. This avoids redundant information being collected. The more information collected, the more precise a query can be generated. Users can decide when to stop asking questions. The system can also determine if enough information has been collected.

The guided search component consists of an intelligent question generator, an inference engine, a question base, a rule base and a predicate-based query composer. In this chapter, the details about how the guided search works are given.

6.1 Question Base

The guided search uses simple questions as carrying tools for acquiring information and knowledge about the requested documents. The question base contains all the questions that can be asked to the user. All the questions are based on the dual model, which consists of the document type hierarchy and the folder organization. As discussed in the search strategy of the knowledge-based document search engine, the information collected through these questions is the key for the search engine to quickly zoom in the document type hierarchy and folder organization, and find the documents efficiently and effectively. The question base can be categorized into two groups: one contains document type related questions and another contains folder organization related questions.

In this section, the detailed descriptions about what questions are contained in the question base and how they can be generated are given. Each question is accompanied by a predicate template, which is used to generate a predicate based on the answer of the question. The correlation between the question and the predicate template determines how a question and a predicate are translated from one to the other.

6.1.1 Question Sub-Base I

The question sub-base I is focused on the document type hierarchy. It contains two questions: one asks for document type, the other for content.

Question1: What is the document type?

Predicate template: Type (FI, "Value")

This question is always the first question that will be asked to the user. It permanently appears in the interface. Each answer will be used to replace "Value" in the predicate template to generate a predicate. A hierarchical drop down list is provided to allow users to choose document type. It contains all the document types, organized according to the document type hierarchy. The main drop down list includes the document types right

under the root in the document hierarchy. Each document type in the main drop down list has its drop down lists that reflect the sub-tree under it, and so on. One or multiple selections are allowed.

Question2: What is the content of the documents?

Predicate template: Attribute (FI, "Value")

Following the first question, the system asks user to describe what the document contains. This question also permanently appears in the interface. This question is provided with a frame template to allow the user to fill values to any attribute in the template. The frame template will be updated accordingly whenever the user changes the answer of the first question. Each attribute/value pair will generate a predicate by replacing "Attribute" and "Value" in the predicate template accordingly.

6.1.2 Question Sub-Base II

The question sub-base II is focused on the folder organization. This group of questions is dynamic. Questions are dynamically added, removed, or changed accordingly when the folder organization is changed. Depending on the conversation scenario, only part of questions in this group will be asked. Each question asks for a piece of knowledge about an object related to the document. Whether or not a question is asked depends on whether it helps to zoom in the folder organization. The questions are organized into question trees. Each question tree contains questions related to a specific object. Each node in a question tree represents a question. A link from a parent node to a child node denotes that the question represented by the parent node should be asked before the one represented by the child node. As an example, Figure 6.2 shows two question trees: one for the owner of document, the other for document itself.

The question sub-base II is generated based on the knowledge base. As discussed in Chapter 2, the knowledge base for a folder organization contains a set of domain

knowledge, one for each kind of object. The domain knowledge consists of a domain organization and a set of property relations. The knowledge base is dynamically generated and updated based on the folder organization [15]. Figure 6.1(a)(b)(c) shows the domain knowledge generated based on the folder organization in Figure 5.4. Figure Figure 6.1(a)(b) shows the domain knowledge of the Owner. Figure 6.1(a) shows the domain organization. The domain Affiliation has a sub-domain Department. Figure 6.1(b) shows the property relations of the domain Department. The domain Department has property (Position, EMPLOYEE), (Position, FACULTY), (Class, STUDENT), (Program, PHD), and (Program, MS). The property relations tell that PHD and MS students are STUDENTs of the Department (Says, Department of CIS) at NJIT, and also FACULTY members are EMPLOYEEs of the Department (Says, Department of CIS) at NJIT. Figure 6.1(c) shows the domain knowledge of the document. Subject of the document can be DATABASE or NETWORK & COMMUNICATION. Also Subject DOCUMENT MANAGEMENT is belong to the Subject DATABASE.

The domain knowledge for an object describes what pieces of knowledge about the object have been used in filing the documents. Therefore, answers of the questions asking for the pieces of knowledge described in the domain knowledge can help the search engine quickly zoom in the folder organization. In the rest of this section, how a question tree is dynamically generated given a domain knowledge is discussed. First of all, a question template is defined as follow, where *property*, *object*, *domain,* and *value* are variables.

Question: What is the *property* of the *object* [in *domain*]

Predicate Template: *property*(*object*, *value*[.*domain*])

Selection list: a list of *values*.

Each question in question sub-base II has a selection list, which is used to provide a list of possible answers. A question is created by replacing the *property* and *object* with the real values. The variable *domain* will be replaced with the real value, which depends

on the conversation context between user and the guided search interface, when the question is asked. The variable *value* will be replaced with the answer of the question to generate a predicate. The process for generating a question tree from a given domain knowledge starts from the root node of the domain organization. Given a domain knowledge DK for object O, a question is generated for each domain in the domain organization and each property name in the property relations. Let D be a domain in the domain organization. A question is created by replacing *property* and *object* with D and O respectively. The selection list contains all the values, also called domain instances, of D. If D is the root of the domain organization, then the part in [...] in the question and the predicate template will be removed. For each property name P in domain D, a question is created by replacing *property* and *object* with P and O respectively. The selection list contains all the values of property P.

The algorithms for generating a question tree for a given domain knowledge is given as follow.

Algorithm 6.1 (*Generate Question Tree*) Let K be a domain knowledge for object O in the knowledge base. Let DO be the domain organization of K. The question tree T corresponding to domain knowledge K is generated as follow:

1 Initialize two queues: QUEUE1 and QUEUE2;

2 Visit the root Dp of DO. Call Algorithm 6.2 to create the question sub-tree Tp for domain Dp. Let Qp be the root of Tp. Initialize T as Tp;

3 Add Dp into QUEUE1, Qp into QUEUE2;

4 While QUEUE1 and QUEUE2 are not empty,

4.1 Let Dp be the first element in QUEUE1, Qp be the first element in QUEUE2. Remove Dp and Qp from QUEUE1 and QUEUE2. Qp is the question created regarding domain Dp.

4.2 For each sub-domain D of Dp, do

4.2.1 Call algorithm 6.2 to create a question sub-tree Td for domain D. Let Q be the root of Td;

4.2.2 Add D into QUEUE1, Q into QUEUE2;

4.2.3 Add Q into T as a child of Qp;

5 Return T.

Algorithm 6.2 (Generate a Question Sub-tree)) Let K be a domain knowledge for object O in the knowledge base, DO be the domain organization of K. Let D be a domain in DO, PR be the property relation of domain D. The following algorithm generates a question sub-tree containing all the questions regarding the domain D and all the properties in D.

1 Initialize question tree T;

2 Initialize two queues QUEUE1 and QUEUE2;

3 If D is empty, create an empty question Q. Otherwise, instantiate a question Q using the question template. Replace *property* and *object* with D and O respectively. Add all the instances of D into the selection list. Add Q as the root of T;

4 Add D into QUEUE1, Q into QUEUE2;

5 While QUEUE1 and QUEUE2 are not empty, do

5.1 Let P be the first element in QUEUE1, Q be the first element in QUEUE2. Remove P and Q from QUEUE1 and QUEUE2;

5.2 For each child Pi of P in PR, do

5.2.1 Let Ni be the property name, Vi be the value. If P is not equal to D and Ni equals to the property name in P, then

- Add Vi into the selection list of Q;
- Add Q into QUEUE2;

Otherwise, if Pi has the same property name as its sibling Pj that has already been visited, let Qj be the question created for Pj, then

- Add Vi into the selection list of Qj;

- add Qj into QUEUE2;

Otherwise,

- Instantiate a question Qi using the question template;
- Replace *property* and *object* with Ni and O respectively;
- Add the value of Vi into the selection list;
- Add Qi into QUEUE2;
- Add Qi into T as a child of Q;

5.2.2 Add Pi into QUEUE1;

6 Return T.

Figure 6.2 shows the question sub-base II generated automatically based on the domain knowledge in Figure 6.1 using the Algorithm 6.1 and Algorithm 6.2. According to the folder organization in Figure 5.4, knowledge of two different kinds of objects is used in document filing. One is the owner (i.e., sender, receiver, author, etc.) of the document. The other is the document itself. So Figure 6.1 shows domain knowledge for owner and document. The question sub-base II in Figure 6.2 contains two question trees. Figure 6.2(a) shows the question tree for owner of documents. Figure 6.2(b) shows the question tree for documents. The relation between two trees can be defined in the rule base, which will be addressed later.

In the remainder of this section, we go through the process of generating the question tree for owner, shown in Figure 6.2(a), to illustrate Algorithm 6.1 and 6.2. The questions in this question tree are numbered for illustration purpose. Algorithm 6.1 started from domain Affiliation. Algorithm 6.2 was called in step 2 of Algorithm 6.1 to create a sub-tree for domain Affiliation.

Question 1, which is regarding domain Affiliation, was generated in step 3 of Algorithm 6.2. Since domain Affiliation has no properties, a sub-tree with a single node (i.e., the question 1) was returned.

In step 3 of Algorithm 6.2, the domain Affiliation and question 1 were added to QUEUE1 and QUEUE2 respectively. In the while loop (step 4), the domain Affiliation and question 1 were removed from QUEUE1 and QUEUE2 in step 4.1. Step 4.2 then visit each child of Affiliation. Algorithm 6.2 was called in step 4.2.1 to generate a sub-tree for domain Department.

Algorithm 6.2 started from the root, which is the domain Department, of the property relation in Figure 6.1(b). Question 2 was created in step 3. The domain Department and question 2 were added to QUEUE1 and QUEUE2 in step 4. In the while loop (step 5), The domain Department and question 2 were removed from QUEUE1 and QUEUE2 in 5.1. Step 5.2 visit each child of Department, which is (Position, EMPLOYEE) and (Class, STUDENT). Question 3 and question 4 were created for (Position, EMPLOYEE) and (Class, STUDENT) respectively. Both were added as children of question 2. When step 5.2 finished, QUEUE1 contained (Position, EMPLOYEE) and (Class, STUDENT). QUEUE2 contained question 3 and question 4. In the second iteration of step 5, (Position, EMPLOYEE) and question 3 were removed from QUEUE1 and QUEUE2 in step 5.1. Step 5.2 visited the child of (Position, EMPLOYEE), which is (Position, FACULTY). Because (Position, FACULTY) has the same property name Position as (Position, EMPLOYEE), no question was generated. Instead, the value FACULTY was added to the selection list of question 3. After step 5.2, QUEUE1 contained (Class, STUDENT) and (Position, FACULTY). QUEUE2 contained question 4 and question 3. In the third iteration of step 5, (Class, STUDENT) and question 4 were removed from QUEUE1 and QUEUE2 in step 5.1. Step 5.2 visited the two children (Program, PHD) and (Program MS). Question 5 was created for (Program, PHD) and added to the sub-tree as a child of question 4. No question was created for (Program, MS) because it has the same property name as its sibling. Instead, the value MS was added to the selection list of question 5. After step 5.2, QUEUE1 contained (Position, FACULTY), (Program, PHD), and (Program, MS). QUEUE2 contained

question 3, question 4 and question 5. Since none of the node in QUEUE1 has child, Algorithm 6.2 returned a sub-tree that contains question 2, 3, 4, and 5, rooted with question 2.

Back to Algorithm 6.1, question 2 was added to the question tree as a child of question 1 in step 4.2.3. This also indirectly linked the other questions under question 2 to the question tree. Because domain Department has no child, Algorithm ended by returning a question tree as shown in Figure 6.2(a).

6.2 Rule Base

The rule base is used to maintain the rules for governing the conversation between user and the guided search interface. The guided search interface has two parts. The fixed part is for the question sub-base I. Two questions in question sub-base I appear in the interface permanently. The dynamic part is for the question sub-base II. Questions from question sub-base II chosen by the inference engine are asked one by one. In this section, the detailed description about the rule base is given.

Rule 1: The questions in question sub-base I appear in the fixed part of the interface permanently. The question about the document type in the question sub-base I is the starting point of guided search. The question for the content of the documents follows.

Rule 2: The conversation in the dynamic part of the interface should not start until the question about document type has been answered.

Rule 3: If the number of answers is greater than the threshold for the question about the document type, then give warning that the query may be too general and provide choices either continuing or switching to document browser.

The general rule 1, 2 and 3 are all related to the questions about the document type and content in question sub-base I. As discussed in Chapter 5, the document type plays the critical role in rapidly narrowing the search space. The question about the

document type is simple and straightforward, which is one of reasons to force it to be the starting point of guided search.

Definition 6.1 (Switching point) A switching point is a question in a question tree, from which the guided search tool switches to another tree for the next question.

Rule 4: The first question that the inference engine chooses from question sub-base II is the question in the root of a question tree, whose object is the first object appears in the filing criteria of the folder organization along the paths from root to leaves.

Rule 4 tells where the dynamic part of the conversation between user and the guided search interface should begin. The rule tells the inference engine to check the folder organization and find the first object that appears in the filing criteria from the root to leaf folders. The conversation should start with the question tree of the object found. This is because the knowledge about the object can help the search engine to zoom in the folder organization to lower level of folders. Therefore, questions regarding the object should be asked first. In real operation, there is no need for the inference engine to dynamically check the folder organization. Instead, the question tree for the object is marked when the question sub-base II was generated.

Rule 5: If question A from a question tree is asked and returned with valid answer, then questions as children of A should be asked next.

Rule 6: If question A in a question tree is asked but returned with no valid answer, no questions in the sub-tree rooted with question A should be asked.

Rule 5 and 6 tells what questions should be asked and in what order depending on the conversation context.

Rule 7: If guided search reaches a switching point, then it leaves the current question tree and moves to the root node in another tree.

Rule 7 tells when the conversation should switch to another a question tree. The switch point is detected by finding two folders F and G, where F is the parent folder of G,

and the filing criteria of folder F and G are regarding difference objects. The switch point tells the inference engine that when the conversation regarding the current object reaches a specific level of detail, further knowledge of object will not help the document retrieval. Therefore, the conversation should switch to another object.

Rule 8: If a question from question sub-base II contains a special attribute, the special attribute should be replaced with a real attribute before the question is asked.

Rule 9: If a question contains a variable *domain*, the variable should be replaced by the answers of questions regarding the domains in which the question is asked.

Rule 8 and 9 are used to finalize the question before they are asked. If a special attribute of documents is used to represent an object, rule 8 says it should be replaced with a real attribute, because the document type is known when the dynamic part of the conversation starts. For example, in the folder organization shown in Figure 5.4, Owner is a special attribute that represents Sender or Receiver of Letter types, Author of publication types. When a question from question sub-base II is asked, the document type should be known. Therefore, the special attribute owner can be replaced with the real attribute. Most questions in question sub-base II contain variable *domain*. Before they are asked, the rule 9 tells how the variable *domain* is replaced with a real value.

6.3 Inference Engine

The inference engine is used to determine which question is the next question to ask. In order to make decisions, it needs to access several resources in the system including the question base, rule base and folder organization. The inference engine will be called by the intelligent question generator, right after the first two questions in the fixed part of the interface have been answered properly. The inference engine uses rule 4, 5, 6, and 7 to determine which question should be asked next depending on the conversation context. If the previous question is a leaf node in the question tree, the inference engine checks the folder organization to see if it is a switching point. If it is switching point, rule 7 applies.

Otherwise, the inference engine notifies the intelligent question generator that no more questions to ask.

6.4 Intelligent Question Generator

The intelligent question generator acts as the central station for guided search. It handles the interactive conversations with the users by asking questions and collecting answers. It calls the inference engine to get the next question and make up the question in a user-friendly way by using rule 8 and 9. It verifies the answers to see if they are valid or not and calls the query composer with the predicate template and valid answers.

6.5 Query Composer

Query composer composes a predicate-based query based on the predicate templates of questions and answers passed by the intelligent question generator. If a single answer is passed, the query composer instantiates a predicate and replaces the "Value" in the predicate template with the answer. If multiple answers are passed, multiple predicates are instantiated and linked with operator 'OR'. Each predicate is in the same format as the predicate template and having "Value" replaced by each of the answers. When the intelligent question generator has no more questions to ask or the user decides to stop answering question, the query composer finalizes the query by linking the predicate of each question with "AND" operator and displays it to the user for verification and modification by option. After this, the query will be submitted to the knowledge-based document search engine to process.

6.6 Example

In this section, an example is given to illustrate how a query can be composed through the guided search. Our search goal is to find the papers that were published in a conference. Assume our knowledge about the papers is that the papers are authored by a

Ph.D. student in CIS department at NJIT, the papers are about document management, and the conference date is in January 2000.

Guided search starts by asking the first question about document type. Being not sure what to type for the document type, the user goes through the drop down list and chooses ProceedingArticles. The query composer creates a predicate Type (FI, ProceedingArticles) using the predicate template for the question. The frame template of ProceedingArticles is then shown in the second question about the content. In the value part for attribute ConferenceDate, the user types "January, 2000". A predicate, ConferenceDate (FI, "01/##/2000"), is created using the predicate template for the second question.

When the dynamic part of the conversation starts, the inference engine is called to choose the first question. Because Owner is the first object appears in filing criteria in the folder organization in Figure 5.4, the question tree in Figure 6.2(a) is chosen as the starting point. So the question 1 is returned according to rule 4. At this point, the document type is already known, which is ProceedingArticles. So the question generator replaces the special attribute Owner in the question with Author. So a question "What is the Affiliation of Author?" is asked. In the selection list, the user sees value NJIT. So the user answers with NJIT, because the user knows that the papers are authored by a Ph.D. student of CIS department at NJIT. A predicate, Affiliation(Author, "NJIT"), is then created. According to rule 5, question 2 is chosen by the inference engine as the next question. At this point, the question generator knows that question 2 is asked in domain instance "NJIT". So a question "What is the Department of Author in NJIT" is asked. After answering "CIS", a predicate, Department (Author, "CIS.NJIT"), is created. The inference engine then chose question 3. Question "What is the Position of Author in CIS" is asked. The selection list contains value "EMPLOYEE" and "FACULTY". The user provides no answer because the user does not know. So the inference engine stops looking at questions under question 3. In this particular case, question 3 has no children

to choose even the user answers it. So the inference engine chooses question 4 as the next. The user is asked "What is the Class of Author in CIS.NJIT", and the user answers with "STUDENT". The query composer knows that this question is asked in domain instance CIS in NJIT. A predicate, Class(Author, "STUDENT.CIS.NJIT"), is then created. Similarly, question 5 is asked next. A predicate, Program(Author, "PHD.CIS.NJIT"), is created after answering "PHD". This is a switch point because the folder CISPhD has a filing criterion Program(Owner, "PHD.CIS.NJIT"), and its child folder PUBLICATION has criterion Type(FI, "PUBLICATION"). According to rule 7, the inference engine switches to question tree in 6.2(b). The user is asked "What is the Subject of Document". The selection list contains values "DATABSE", "DOCUMENT MANAGEMENT" and "NETWORK & COMMUNICATION". So the user answers with "DOCUMENT MANAGEMENT". A predicate, Subject(FI, "DOCUMENT MANAGEMENT"), is created. Since no other questions found in the question tree. The inference engine suggests the question generator to end the conversation. Finally, the query composer put all the predicates created during the conversation together and composes the following query:

Type (FI, "ProceedingArticles") ∧

ConferenceDate (FI, "01/##/2000") ∧

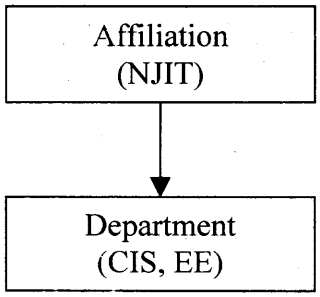
Affiliation (Author, "NJIT") ∧

Department (Author, "CIS.NJIT") ∧

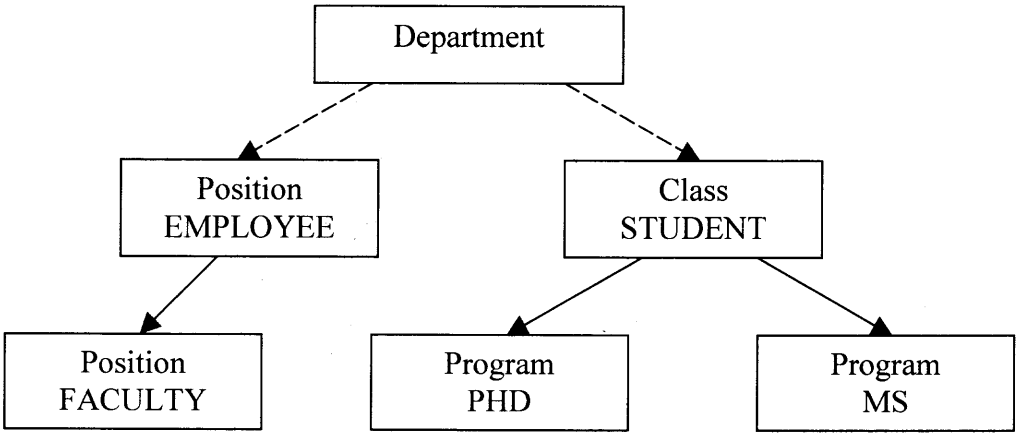
Class (Author, "STUDENT.CIS.NJIT") ∧

Program (Author, "PHD.CIS.NJIT") ∧

Subject (FI, "DOCUMENT MANAGEMENT")



(a) Domain organization



(b) Property relationships of domain "Department"

Figure 6.1 (a)(b) Domain knowledge of the "Owner"

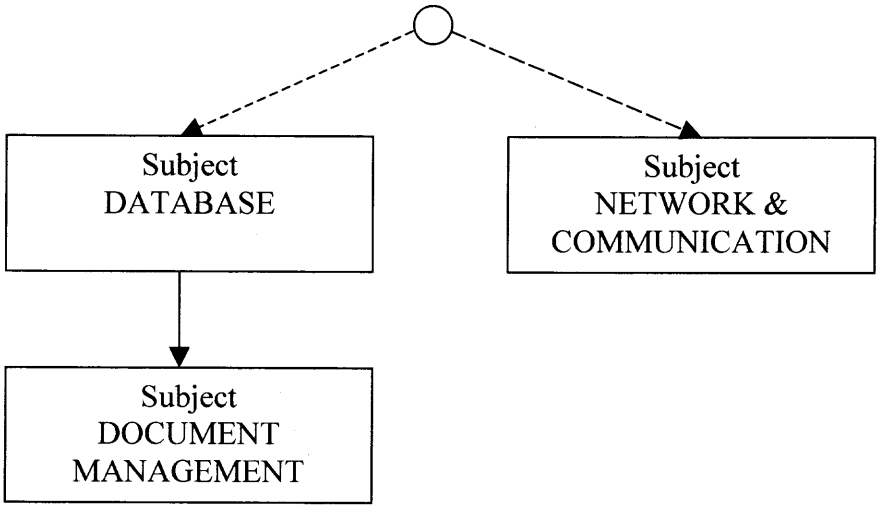
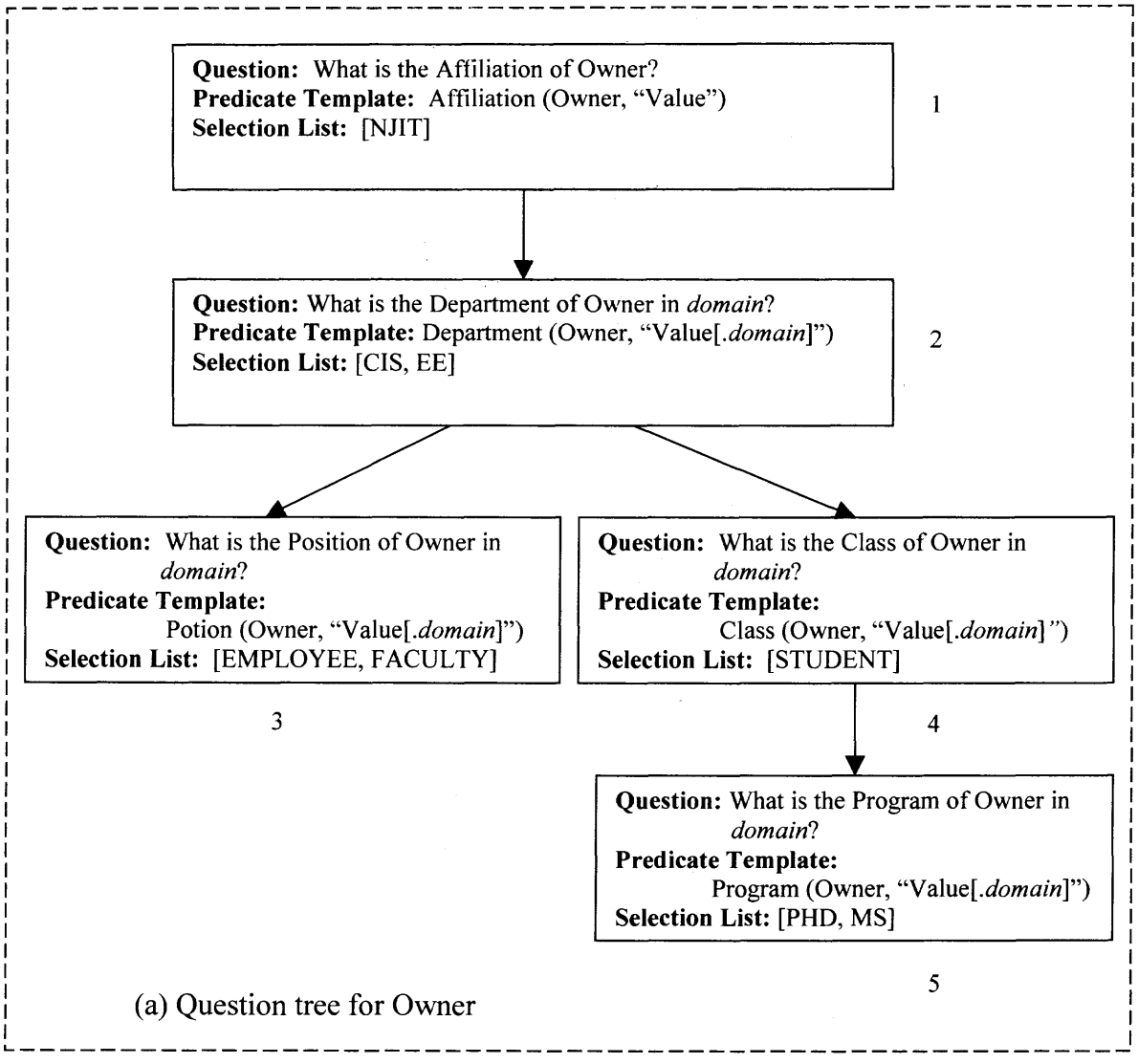


Figure 6.1 (c) Knowledge of the "Document"

Figure 6.1 Knowledge base of the example



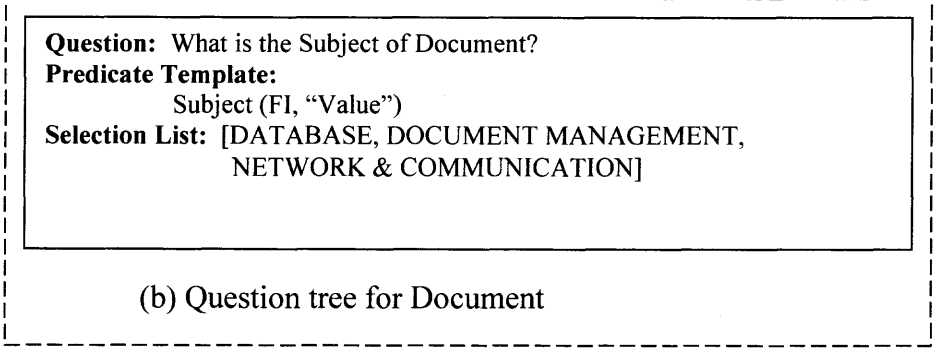


Figure 6.2 Question sub-base II of the example

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This dissertation presents a knowledge-based document retrieval architecture with application to TEXPROS. The architecture is based on a dual document model that consists of a document type hierarchy and a folder organization. This architecture provides two kinds of user interfaces. With the predicate-based query language, experienced users can more precisely and accurately specify the search criteria and their knowledge about the documents to be retrieved. To assist new users formulate queries, a guided search was presented as part of an intelligent user interface. The guided search component includes a question base, a rule base, a question generator and an inference engine, a predicate-based query composer, as well as the guided search user interface. The guided search collects information by asking simple questions about the documents the user wishes to locate. The intelligent question generator and inference engine will generate subsequent questions from the question base depending on the user's answers to previous questions using rules from the rule base. The predicate-based query composer is used to generate a particular query for the user using predicate-based query language after the user finish answering questions. The query can then be displayed to the user for evaluation and modification.

The retrieval architecture provides a query optimizer for normalizing and optimizing the user queries. A knowledge-based search engine was presented for quickly identifying documents that satisfy the user query. Taking advantage of the dual model, the multi-level storage architecture as well as the knowledge base generated during document filing, the search engine can process user queries very efficiently and effectively.

The predicate-based query language was provided for users to search for the documents stored in the system. With this document retrieval language, users can specify both searching criteria and their knowledge about the documents to be searched for more precisely and accurately than simple keyword search. In designing this document retrieval language, much considerations and efforts were given to the question of how to balance between the language's simplicity and its expressive power. While the language has to be sophisticated enough in order to support effective document retrieval, the more complex the language is, the more difficult it is to use and the less efficient it will be to process. The goal of designing the document retrieval language is to preserve the language's simplicity without loosing its necessary expressive power for formulating a precise query specification. To keep consistent with document filing criteria, the presented document retrieval language was based on the general First Order Predicate Logic (FOPL). Certain modifications were made to tailor it for document retrieval purpose.

The knowledge-based query processing includes query optimization and knowledge-based document search. The query optimizer validates each element in the query and normalizes the query into disjunctive normal forms. Predicates in each disjunctive normal form are re-ordered to speedup the retrieval. The knowledge-based search engine processes the query and returns those documents that satisfy the search criteria. To achieve high efficiency, a knowledge-based query preprocessing is performed first to reduce the search space to a small set of documents. The dual model and the multi-level storage architecture make it possible to generate a much smaller and complete set of relevant documents. There are two entries to reduce the search space. One is from document type hierarchy. Identifying the document type enables the search to be concentrated on documents of that type only. The other entry is from folder organization. The search space is reduced by concentrating the search on a specific folder. The search space can further be reduced by focusing on a specific document type in a specific folder.

The key issue is how to efficiently identify the smallest folder that contains all the relevant documents. The process requires the help of the knowledge base and the predicate evaluation engine. The algorithms were given and theoretically proved. Performance analysis was also given.

After the search space is reduced to a small set of documents, documents will be matched against the query. Only the documents that satisfy the search criteria will be returned to users. This match process is needed because the preprocessing only guarantees that the set of documents contains all the candidates that may satisfy the user query. But, not all documents in the set are expected by the user. For the first level predicates in the query, the match process is done by matching the content of the documents with the query. For the second level predicates, the match process needs to call the predicate evaluation engine for determining whether a document satisfying a predicate. So the match process can be content-based or knowledge-based depending on the search criteria.

The guided search was presented as an intelligent search tool for document retrieval. Through guided search, a predicate-based query can be automatically generated after answering a few questions. One of the purposes of a guided search is to help new users who are not familiar with the query language and who know little about the system. The experienced users can also use the guided search as a tool to generate an initial query and then compose the final query based on it. With the assistance of guided search, the experienced users can more easily specify their explicit search goals and minimize the spelling errors.

Another reason of a guided search is to improve the efficiency and effectiveness of document retrieval. In a database, how fast a SQL query can be returned not only depends on database tuning, but also the query itself. An efficient query can take advantage of the database tuning, and therefore, help the database engine give a faster return. The efficiency of knowledge-based document retrieval depends on the

information collected from the users. Taking advantage of the knowledge base collected during document filing, the guided search can tell users what information is most important to quickly find the documents.

Guided search collects information from user by asking simple questions, with the most important ones first. The following questions will depend on the user's answer to the previous question. This avoids redundant information being collected. The more information collected, the more precise the query can be generated. Users can decide when to stop asking questions. The system can also determine if enough information has been collected.

Algorithms for dynamically generating the question base for a folder organization were given. Rules are defined for governing the conversation between the user and the guided search interface. Examples were given to illustrate how the question base was generated and how guided search works.

The main contribution of this dissertation is the knowledge-based document retrieval methodology that supports more precise queries without sacrificing efficiency and ease of use. With the dual model, knowledge about how documents are organized in office environment, as well as the conceptual structures of documents, are used in document retrieval. The predicate-based language gives users more freedom to specify what they know about the requested documents. So the search engine understands more precisely what users want. This is a necessary requirement for improving the effectiveness of document retrieval. The knowledge-based algorithms provide an efficient solution for processing queries in the predicate-based language. For easy of use, a guided search was developed to help users compose queries.

As a future work, the knowledge-based document retrieval methodology should be implemented and integrated it into TEXPROS. As another direction of future work, a more user-friendly language with the same expression power as the predicate-based language should be investigated. So users can write queries more easily without having to

use the guided search. A more friendly language will encourage users to provide more information they know about the desired documents, and therefore, help the search engine find the documents more quickly. Finally, effort should be made to improve the guided search interface. The guided search uses a set of dynamically generated simple questions to collect information from user. The questions can be improved so users can understand them more easily. A new style for the conversation between the user and the system can also be investigated to make the interface more friendly.

REFERENCES

1. N. Adami, A. Bugatti, A. Corghi, R. Leonardi, P. Migliorati, L. A. Rossi and C. Saraceno, "ToCAI: a Framework for Indexing and Retrieval of Multimedia Documents", Proceedings of the 10th International Conference on Image Analysis and Processing, Venice, Italy, pp.1027-1032, September 1999.

2. E. Appiani, L. Boato, S. Bruzzo, A.M. Colla, M. Davite and D. Sciarra, "STRETCH: A System for Document Storage and Retrieval by Content", Proceedings of the 10th International Workshop on Database & Expert Systems Applications, Florence, Italy, pp. 588-592, September 1999.

3. R. Baeza-Yates and G. Navarro, "Block Addressing Indices for Approximate Text Retrieval", Journal of the American Society for Information Science, vol. 51, no. 1, pp. 69-82, 2000.

4. E. Bertino, B. Catania, B. Black, J. McNaught, F. Rinaldi, A. Brasher, D. Deavin, A. Persidis, V. Candela, F. Esposito, G. Semeraro and G. P. Zarri, "CONCERTO, Conceptual Indexing, Querying and Retrieval of Digital Documents", Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, vol. 2, pp. 1106 - 1109, June 1999.

5. A. Celentano, M. Fugini, and S. Pozzi, "Knowledge-Based Document Retrieval in Office Environments: The Kabiria System", ACM Transactions on Office Information Systems, vol. 13, no. 3, pp. 237--268, July 1995.

6. C. Chang and C. Hsu, "Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 4, pp. 595-609, July/August 1999.

7. H. Chen, "Knowledge-Based Document Retrieval: Framework and Design", Journal of Information Science: Principles & Practice (Amsterdam), vol. 18, no. 4, pp. 293-314, 1992.

8. S. Chen and Y. Horng, " Fuzzy Query Processing for Document Retrieval Based on Extended Fuzzy Concept Networks", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 29, no. 1, pp. 96-104, 1999.

9. S. Chen and J. Wang, "Document Retrieval Using Knowledge-Based Fuzzy Information Retrieval Techniques", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, no. 5, pp. 793-803, May 1995.

10. W. W. Chu, C. C. Hsu, A. F. Cárdenas, and R. K. Taira, "Knowledge-Based Image Retrieval with Spatial and Temporal Constructs", IEEE Transactions on

Knowledge and Data Engineering, Vol. 10, No. 6, pp.872-888, November/December 1998.

11. J. F. Cullen, J. J. Hull and P. E. Hart, "Document Image Database Retrieval and Browsing Using Texture Analysis", in Proceedings of the 4th International Conference Document Analysis and Recognition (ICDAR '97), Ulm, Germany, vol. 2, pp. 718-721, August 1997.

12. M. Cutler, H. Deng, S. S. Maniccam, W. Meng, "New Study on Using HTML Structures to Improve Retrieval", in Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI '99), Chicago, IL, USA, pp. 406-409, November 1999.

13. S. Doong, C. Wei, X. Fan, D. C. Hung and P. A. Ng, "A Folder Organization Model in the Office Environment", in Proceedings of the 4th International Conference on Information Systems Analysis and Synthesis, 1998.

14. Y. Dong, A More Efficient Document Retrieval Method for TEXPROS, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, January 2001.

15. X. Fan, Knowledge-Based Document Filing for TEXPROS, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, May 1998.

16. X. Fan, F. Sheng, X. Li, Z. Cheng and P. Ng, "A Scalable Automated System for Document Management", in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, Dallas, Texas, June 2000.

17. X. Fan, Q. Liu and P. Ng, "An Automated Document Filing Systems", Journal of Systems Integration, Vol. 9, No. 3, pp. 223-262, 1999.

18. X. Fan, F. Sheng, P. Ng, "DOCPROS: A Knowledge-based Personal Document Management System", the 10th International Workshop on Database and Expert Systems Applications (DEXA'99), Florence, Italy, pp. 527-531, September 1999.

19. X. Fan, F. Sheng, S. Doong, P. Ng and C. Wei, "A Process for Constructing a Personal Folder Organization", in Proceedings of International Workshop on Multimedia Database, Dayton, Ohio, pp. 20 – 27, August 1998.

20. L. Gravano, H. García-Molina and A. Tomasic, "GIOSS: Text-Source Discovery Over the Internet", ACM Transactions on Database Systems, vol. 24, No. 2, pp. 229-264, 1999.

21. X. Hao, Automatic Office Document Classification and Information Extraction, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, August 1995.

22. Y. He, Z. Jiang, B. Liu and H. Zhao, "Content-Based Indexing and Retrieval Method of Chinese Document Images", in Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR'99), Bangalore, India, pp. 685-688, September 1998.

23. J. Horng and C. Yeh, "Applying Genetic Algorithms to Query Optimization in Document Retrieval", Information Processing and Management, vol. 36, no. 5, pp. 737-759, 2000.

24. J. Hu, X. Li, S. Dong, D.C. Hung and P.A. Ng, "A Thesaurus Model for Document Processing System: A TEXPROS Approach", in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, Dallas, Texas, June 2000.

25. J. Hu, Knowledge Management for TEXPROS, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, May 1999.

26. S. C. Hui and A. Goh, "Incorporating Fuzzy Logic with Neural Networks for Document Retrieval", Engineering Applications of Artificial Intelligence, vol. 9, no. 5, pp. 551-560, 1996.

27. P. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel and Z. Protopapas,"Fast and Effective Retrieval of Medical Tumor Shapes", IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 6, pp. 889-904, November/December 1998.

28. P. Lambrix and N. Shahmehri, "Towards Creating a Knowledge Base for World-Wide Web Documents", Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems, Grand Bahama Island, Bahamas, pp. 507-511, December 1997.

29. X. Li, Z. Cheng, F. Sheng, X. Fan and P. Ng, "A Document Classification and Extraction System with Learning Ability", in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, Dallas, Texas, June 2000.

30. X. Li, J. Hu, X. Fan, C. Y. Wang and P. A. Ng, "Automated Document Filing and Retrieval System: An Overview", in Proceedings of the Third Biennial World Conference on Integrated Design and Process Technology, Vol. 4, Berlin, Germany, pp. 231-241, July 1998.

31. X. Li, Automatic Document Classification and Extraction System (ADoCES), PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, May 1999.

32. J. H. Lim, "Learning Visual Keywords for Content-Based Retrieval", Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, vol. 2, pp. 169-173, June 1999.

33. J. Lin, Collaborative Software Agents Support for the TEXPROS Document Management System, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, January 2000.

34. J. Lin, H. Shen, T.M. Chu, S. Doong, R.Curtis, D.C. Hung and P.A. Ng, "Folder Organization Query Language", in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, Dallas, Texas, June 2000.

35. Q. Liu, An Office Document System With the Capability of Processing Incomplete and Vague Queries, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, August 1994.

36. Q. Liu and P. Ng, "A Browser of Supporting Vague Query Processing in an Office Document System", Journal of Systems Integration, vol. 5, no. 1, pp. 61-82, 1995.

37. Q. Liu and P. Ng, Document Processing and Retrieval: Text Processing, Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

38. R. Marega and M. T. Pazienza, "CoDHIR: An Information Retrieval System Based on Semantic Document Representation", Journal of Information Science, vol. 20, no. 6, pp. 399-412, 1994.

39. P. Martin and P. W. Eklund, "Knowledge Retrieval and the World Wide Web", IEEE Intelligent Systems, Vol. 15, No. 3, pp. 18-25, May/June 2000.

40. B. P. McCune, R. M. Tong, J. S. Dean and D. G. Shapiro, "RUBRIC: A System for Rule-Based Information Retrieval", IEEE Transactions on Software Engineering, vol. SE-11, no. 9, 1985.

41. M. Mechkour, P. Mulhem, F. Fourel, and E. F. C. Berrut, "PRIME-GC: A medical information retrieval prototype on the Web", in Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE '97), Birmingham, UK, pp. 2-9, April 1997.

42. R. M. Rohrer, J. L. Sibert and D. S. Ebert, "A Shape-Based Visual Interface for Text Retrieval", IEEE Computer Graphics and Applications, Vol. 19, No. 5, pp. 40-46, September/October 1999.

43. P. O'Neil, "An Incremental Approach to Text Representation, Categorization, and Retrieval", in Proceedings of the 4th International Conference Document Analysis and Recognition (ICDAR '97), Ulm, Germany, vol. 2, pp. 714-717, August 1997.

44. M. Ortega-Binderberger, S. Mehrotra, K. Chakrabarti and K. Porkaew, "WebMARS: A Multimedia Search Engine", in Proceedings of SPIE - The International Society for Optical Engineering Proceedings of the 2000 Internet Imaging, San Jose, CA, USA, pp. 314-321, January 2000.

45. E. Ozkarahan, "Multimedia Document Retrieval", Information Processing & Management, vol. 31, no. 1, pp. 113-131, 1995.

46. U. Schiel, I. M. S. F. Sousa and E. Ferneda, "SIM - A System for Semi-Automatic Indexing of Multilingual Documents", in Proceedings of the 10th International Workshop on Database & Expert Systems Applications, Florence, Italy, pp. 577-581, September 1998.

47. H. Shen, J.T. Lin, T.M. Chu, M.M. Tanic, R.Curtis, D.C. Hung and P.A. Ng, "Automatic Authoring in HyTEXPROS," in Proceedings of the Fifth World Conference on Integrated Design and Process Technology, Dallas, Texas, June 2000.

48. H. Shen, HyTEXPROS: A Hypermedia Information Retrieval System, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, January 2000.

49. A. F. Smeaton and A. L. Spitz, "Using Character Shape Coding for Information Retrieval", in Proceedings of the 4th International Conference Document Analysis and Recognition (ICDAR '97), Ulm, Germany, vol. 2, pp. 974-978, August 1997.

50. P. J. Smith, S. J. Shute, D. Galdes and M. H. Chignell, "Knowledge-Based Search Tactics for an Intelligent Intermediary System", ACM Transactions on Office Information System, vol. 7, no. 3, 1989.

51. B. W. Stalcup, P. W. Dennis and R. B. Dydyk, "Automated Search and Retrieval of Information from Imaged Documents Using Optical Correlation Techniques", in Proceedings of SPIC - The International Society for Optical Engineering Proceedings of the 1999 Algorithms, Devices, and Systems for Optical Information Processing III, Denver, Co, USA, pp. 92-101, July 1999.

52. D. Skuce, "Integrating Web-Based Documents, Shared Knowledge Bases, and Information Retrieval for User Help", Computational Intelligence, vol. 16, no. 1, pp. 95-113, 2000.

53. C. Wang, An Intelligent Browser for TEXPROS, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, May 1998.

54. C. Wei, Knowledge Discovering for Document Classification Using Tree Matching in TEXPROS, PhD dissertation, Department of Computer and Information Science, New Jersey Institute of Technology, Newark, New Jersey, May 1996.

55. C. Wei, Q. Liu, J. Wang, and P. Ng, "Knowledge Discovering for Document Classification Using Tree Matching in TEXPROS", Information Sciences, vol. 100, no. 1-4, pp. 255--310, August 1997.

56. L. Wilcox and J. Boreczky, "Annotation and Segmentation for Multimedia Indexing and Retrieval", Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS'98), Kohala Coast, HI, vol. 2, pp. 259-266, January 1998.

57. Z. Zhu, Q. Liu, J. McHugh, and P. Ng, "A Predicate Driven Document Filing System", Journal of Systems Integration, vol. 6, no. 3, pp. 373-403, September 1996.

58. Z. Zhu, J. McHugh, J. Wang, and P. Ng, "A Formal Approach to Modeling Office Information Systems", Journal of Systems Integration, vol. 4, no. 4, pp. 373-403, December 1994.