INFORMATION RETRIEVAL AND ANSWER EXTRACTION FOR AN XML KNOWLEDGE BASE IN WEBNL

By

WILASINI PRIDAPHATTHARAKUN

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2001

To my Parents and Brothers

ACKNOWLEDGMENTS

Without the assistance, priceless advice, and the effort and patience of many people, this study could not have been completed. I would like to take this opportunity to thank all of them very much.

I want to especially honor Dr. Douglas D. Dankel II, a faculty member of the Computer and Information Science and Engineering (CISE) Department at the University of Florida (UF) and supervisor of this thesis. Despite being very busy, he was willing to provide his precious time. He guided and encouraged me through this study, and also gave me his valuable advice and warm understanding. I appreciate his effort and patience with me while I made my way to the completion of this research. I also give thanks and praise to my supervisory committee members, Dr. Joachim Hammer, Dr. Sanguthevar Rajasekaran, and Dr. Ralph Selfridge who provided their valuable time to give scholarly comments concerning this study.

I am very grateful to the faculty of CISE at UF especially the professors with whom I studied. All of them gave me a precious experience as a graduate student here. My sincere thanks and appreciation go to John Jeffery Bowers, the graduate secretary of CISE at UF and other Administrative Staff who gave me many useful suggestions until the day of my graduation.

I am very proud to be a part of the WebNL development team and to work with my colleagues, Eugenio Jarosiewicz, Nathaniel Nadeau, and Nicholas Antonio. Many

thanks go to the Thai students at UF for their friendship and assistance that helped me throughout my studies at UF.

Finally, my wholehearted thanks go to all of the members of my family in Thailand. They always gave me their warm care, encouragement, and financial support through out my studies at UF.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

INFORMATION RETRIEVAL AND ANSWER EXTRACTION FOR AN XML KNOWLEDGE BASE IN WEBNL

By

Wilasini Pridaphattharakun

December 2001

Chairman: Douglas D. Dankel II
Major Department: Computer and Information Science and Engineering

Searching for information from any of the existing knowledge bases on the web is very fashionable. However, the large-scale web search engines are often unable to retrieve the desired information of interest to the users. This is because of the amount of information on the web significantly increasing every day (requiring these search engines to continually update their indexes), and using a set of unordered keywords often results in a significant number of the retrieved pages that are not relevant. The WebNL project at the University of Florida aims to develop a system that gives high quality answers to queries posed by users in natural language. This thesis is a part of the WebNL project. The purpose of this research is to create an Information Retrieval and Answer Extraction (IR and AE) module to retrieve a precise answer from a knowledge base for a user. To achieve this goal, the system uses the following three distinct phases: Question Analyzing, Element Indexing, and Answer Generating. The contribution of this research

is to make information searching on CISE graduate web pages of the University of Florida more efficient.

CHAPTER 1
INTRODUCTION

Currently, searching for information from any of the existing knowledge bases on the web is very fashionable. Users can find the information they desire by typing an unordered set of keywords. However, the large-scale web search engines are often unable to retrieve just the desired information of interest to the users. This is due to two problems: the amount of information on the web is significantly increasing every day (requiring these search engines to continually update their indexes) and using a set of unordered keywords often results in a significant number of the retrieved pages that are not relevant.

Researchers have been developing high performance systems, Question Answering (QA) systems, to solve these problems [HOVY2000, MILW2000]. Using a combination of advanced approaches (i.e., Natural Language Parsing (NLP), Information Retrieval (IR), and Information Extraction (IE)), QA systems retrieve the best possible answer, which is related to a user's natural language query, from the knowledge base.

The goal of the WebNL project at the University of Florida is to develop a system that gives high-quality answers to queries posed by users in natural language (e.g., English). The techniques used include Natural Language Parsing (NLP), Information Retrieval (IR), Answer Extraction (AE), Extensible Markup Language-Knowledge Base (XML-KB) Representation, and Natural Language Generation (NLG).

Overview of the System

The WebNL system (Figure 1) was developed for understanding and answering requests from users expressed in natural language. The system acquires a question from a user in English and then uses QA techniques to generate an answer for the user through the Graphic User Interface (GUI). The system is organized into four main modules—XML-KB, NLP, IR and AE, and NLG.

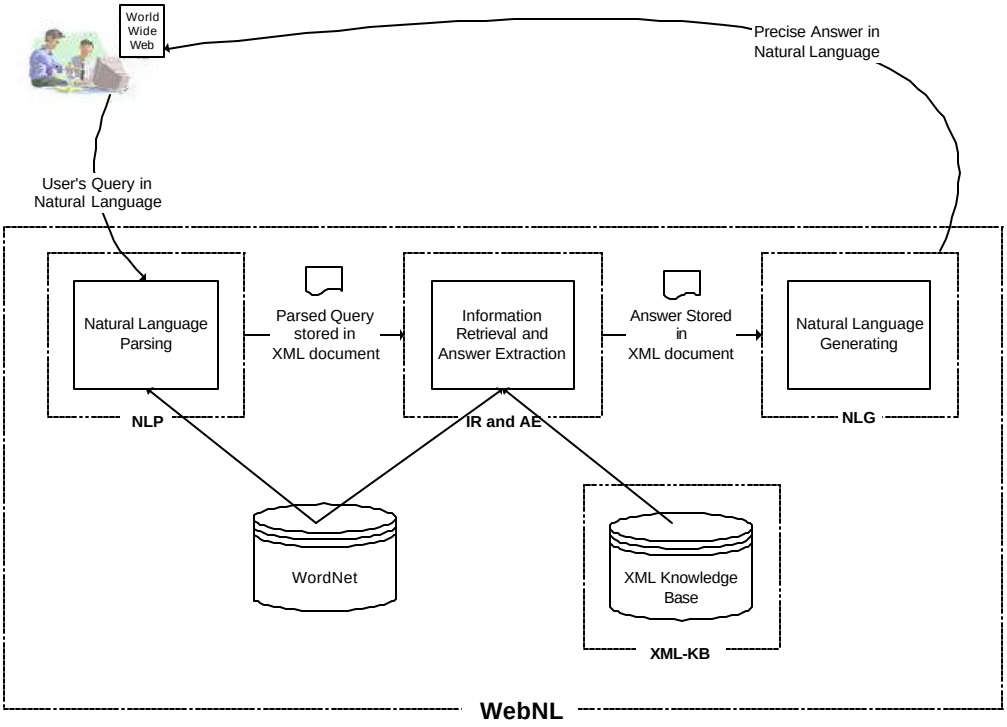

Figure 1-1. Overview of WebNL system.

In the first module (XML-KB) the knowledge of the domain is represented in a knowledge base using the Extensible Markup Language (XML) [W3C1998] based on a meta-data language representation. This representation defines the way to express information via our own customized markup language for different classes of documents. The second module (NLP) analyzes a natural language question producing a parse tree from the parts of speech defined for each word. Then, the IR and AE module construct a query from the parse tree to find and retrieve the correct information from the XML

knowledge base. The result from the IR and AE module is a well-formed answer written in XML. The last module (NLG) takes this XML document from the XML-KB module and transforms it into Natural Language, which is displayed to the user.

This thesis focuses on the IR and AE module. This module uses IR techniques to retrieve possible results from the XML knowledge base. The input to this module is an XML file containing the analyzed structure of the user's query. This structure is processed by the system to generate a query that retrieves an answer from the knowledge base. To produce a more precise answer, AE techniques are used to extract the most relevant answer from the results returned by the IR techniques. However, AE techniques are expensive. Linguistic knowledge and several methods, such as question processing and pattern matching, are needed to determine the semantics of a query and the semantics of the information in the XML knowledge base. A more detailed discussion of this process is given in the following section.

Purpose of this Research

The purpose of this research is to create an IR and AE module (see Figure 2) to retrieve a precise answer from a knowledge base for a user. To achieve this goal, the system employs the following three distinct phases: Question Analyzing, Element Indexing, and Answer Generating.

In the Question Analyzing phase, the system uses linguistic knowledge to classify the type of the user's request and the expected answer type. For example, a "what" question is an informative question and requires an informative answer, while a "where" question involves a location answer. To answer a question the system locates keywords within the head noun, main verb, and adjective phases from the question. For example, if the question posed is "How many credits do thesis students have to obtain to graduate?,"

the head noun of the question is “credits,” the main verb is “obtain,” and the keywords are “credits, thesis students, obtain, graduate.” The system also attempts to find synonyms to the head noun, the main verb, and the key words. The results of Question Analysis (i.e., the type of question, the type of expected answer, the head noun, the main verb, and the keywords) are place into a customized pattern, which is used in the following processing phases: Element Indexing and Answer Generating.

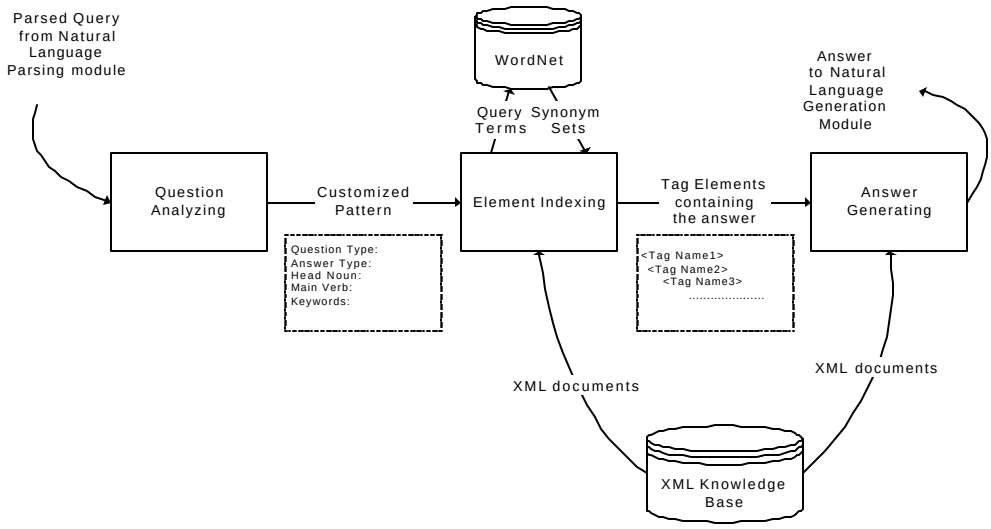

Figure 1-2. Overview of IR and AE module.

Element Indexing tries to collect XML documents containing the answer. To locate an answer, the system uses two techniques (Directory Searching and Tag Element Searching) to search every XML document comparing the document’s semantic tag elements to the terms embedded in the pattern generated from Question Analyzing. All tag elements containing the answer are indexed and sent to the next phase, Answer Searching. A keyword matching technique is used as a third approach in case no tag elements are indexed using the two techniques above. In this third approach, the matcher searches all of the content of every XML document finding the most occurrences of the term(s) generated from Question Analyzing. The tag elements containing the answer are indexed and used for the next phase, Answer Searching. The system also uses a scoring

method for these three techniques to score possible answers guided by the head noun and the keywords extracted by Question Analyzing. The answer with the largest score is selected to be the correct answer. In addition, WordNet [MILL1998] is used to find synonyms of the terms with the aim of improving the performance of this searching.

A simple and powerful query language, XML Query Language (XQL), performs Answer Generating. The tag elements returned from Element Indexing are used to construct a query expressed in XQL to retrieve an answer from the XML knowledge base.

This thesis is organized as follows. First, the relevant literature on IR and AE are reviewed in Chapter 2. Chapter 3 presents background about XML, XQL, and WordNet, which are used in the WebNL system. An overview of the system architecture and the research methodology used for this thesis is described in Chapter 4. Chapter 5 gives examples of answer searching to natural language requests. Finally, Chapter 6 gives conclusions, contributions and limitations of the research, and suggestions for further study.

CHAPTER 2
RELATED RESEARCH

Information Retrieval (IR) techniques are successful for locating a large number of documents containing the answer of a user's query. However, the user requires a correct answer to his/her question instead of a whole document that must then be further searched. The Question Answering (QA) system attempts to tackle this problem by extracting the document content in more depth. To reduce documents and find a more precise answer, Hull [HULL1999], Ferret et al. [FERR2000], and Moldovan et al. [MOLD1999] introduce some interesting strategies involving Parsing, Question Analysis, Proper Name Recognition, Query Formation, and Answer Extraction (AE). Research related to QA systems has increased over the past few years with the growth of information on the Web. This chapter presents a summary of some of this research, which lead to the creation of the WEBNL system that my colleagues and I have developed.

English-Language Question Answering System for a Large Relational Database

Waltz [WALTZ1978] created the system called Programmed LANguage-based Enquiry System (PLANES). PLANES is a question answering system, which gives the user an explicit answer to a natural language request for information from the U.S. Navy 3-M (Maintenance and Material Management) database of aircraft maintenance and flight data. The request, a sentence, is first parsed using parsing and grammar verifying techniques. The system then tries to generate a query from the parsed representation of the sentence to retrieve an answer from the relational database. Finally, the retrieved

answer is displayed to the user using a selected style. The system is divided into four main tasks: parsing, query generation, evaluation, and response generation.

Parsing

An input question is first verified and corrected to ensure that all phrases and words of the query are spelled correctly. The system attempts to replace all phrases and words with appropriate forms. The system then matches and parses phrases with their related phrase patterns called subnets. These subnets also store the parsed phrases in a canonical form using context registers. The context registers serve as a history keeper, tracking values from previous questions. Using this information, the system can resolve missing information and pronoun references occurring in the current question. Noise words from the user query, such “please tell me” and “could you tell me,” are eliminated. Using the canonical values stored in the context registers, the system generates a provisional query that is used in the next module, the query generation.

Query Generation

In the query generation phase, the provisional query developed by the parser is converted into a formal query. The system attempts to decide which relations, input fields, output fields, operations, and constant values should be used in the actual query. Using a relational calculus expression, the formal query is then constructed. To ensure that the system understands the request from the user, the system creates a meaning paragraph from the formal query, which is returned to the user for approval.

Evaluation

In the evaluation phase, the system uses the formal query expression to retrieve an answer from the relational database. The system first selects the files that are to be searched. The order for searching the files is determined. The system then performs the

search to obtain the results. The results from different relations are combined to obtain the precise answer. Finally, the system saves the results for further use.

Note that the system is able to process sophisticated requests, using multiple clauses and comparatives. The system processes the modifying phrases, clauses, or comparatives as a normal request before considering the actual request in order to find the boundary of search for the user's answer.

Response Generator

The response generator phase translates the output from searching the database into a simple number/a list of numbers, a graph, or a table depending on the requested style or what the system determined to be the most appropriate form.

Question Answering in Webclopedia

Hovy, Gerber, Hermjakob, Junk, and Lin [HOVY2000] propose a system, called Webclopedia. The system accepts a question from the user, uses the parser-based approach to analyze the question's text, applies IR techniques to retrieve documents containing an answer, then uses word-level and syntactic/semantic-level techniques to return the specific answer to the user. The QA processes are described as follow.

Parsing

The CONTEX parser, originally generated by Hermjakob and Mooney [HERM1997], is used to find the semantics of the user's question. The parser annotates the structure of the question, (i.e., phrases, nouns, verb phrases, and adjectives), then the parsed question is marked with QA information including the semantic type of the question, the semantic type of the desired answer (which Hovy et al. [HOVY2000] call the QTARGET), a main head noun (called QARGS [HOVY2000]), and other keywords.

Retrieving and Ranking Documents

Unlike the PLANES system, the Webclopedia system does not use a relational database as its knowledge base. The information is retained as documents. The major keywords used in a question form the question terms used to create a query for document retrieval. To improve the document retrieval, the system expands the query with synonyms of the question terms by using WordNet1.6 [FELL1998], an on-line network of semantically related words and terms. A search engine, called MG developed by Witten [WITT1994], is then used to retrieve the documents. The system specifies the threshold for the number of documents returned by MG. Two techniques, relaxing query terms and dropping query expansion, are employed to increase the number of returned documents and to decrease the number of returned documents, respectively.

Because a large number of documents are typically retrieved, the retrieved documents are ranked by a scoring method so the top 1000 can be selected. The scoring method assigns a score to each document, based on the number of question term occurrences in each document, and the types of those terms (i.e., question terms or synonyms). For example, according to Hovy et al. [HOVY2000], "each word in the question gets a score of 2, each synonym of each word gets a score of 1, and other words get a score of 0." The total score of each document is calculated by the formula, "Document score = sum of word scores / number of different words" [HOVY2000].

Segmenting Document and Ranking Segment

The system segments the selected documents to focus on determining a precise answer by using TextTiling developed by Hearst [HEAR1994] and C99 developed by Choi [CHOI1999]. Each document is partitioned into smaller segments where the

answers might be located. The same scoring method is used to rank the segments. The topmost 100 segments are chosen to find the precise answer.

QA Typology

Unlike the PLANES system, Webclopedia uses a QA typology to match a user question, not subnets. Hovy et al. [HOVY2000] build the QA typology, a catalog of QA types that cover all forms of simple questions and answers. The QA typology consists of typical patterns of expressions in terms of QTARGET and QARGS of both questions and answers. The system tries to assign an appropriate pattern of QA typology to a parsed query.

Answer Matching

To identify the answers, the matcher first matches the chosen QA pattern against the parsed query and the text segments. If the matching fails to obtain the answers, the system then uses a specified function to determine the answer by scoring the position of words in each text segment. The text segment having the highest score is selected as the final answer.

Xerox TREC-8 Question Answering Track Report

Another interesting system for Question Answering and Natural Language Processing (NLP) is called the Xerox TREC-8 question answering system developed by Hull [HULL1999]. The system is designed to accept a user's question and returns a precise answer. A parser, effective at finding the semantics of a question, transforms the question into a structured query. An IR technique expressed in the query's terms retrieves documents in which the answer is located. Partitioning the top ranked documents into sentences, including tagging proper nouns to words in the sentences, leads the system to develop the answer. The system is composed of five main distinct

methods: question parsing, sentence boundary identifying, sentence scoring, proper noun tagging, and answer presentation.

Question Parsing

Question parsing in this system is similar to that of Webclopedia system. That is the question is parsed and tagged for parts of speech. The parsed question is categorized based on the question type to generate the semantic type of expected answer: a person, place, time, money, quantity, and number.

Sentence Boundary Identifying

Hull [HULL1999] utilizes an IR system, called the AT&T's TREC-7 adhoc system, provided by Amit Singhal to retrieve documents using the question terms. Each top ranked documents is divided into sentences using sentence boundaries, such as "?", ")", and ".".

Sentence Scoring

Each sentence is scored on the basis of the number of query terms and type of the query terms found in that sentence. The system selects the topmost scoring sentences to continue searching for the answer.

Proper Noun Tagging

The sentences, which are selected by the sentence scoring module, are tagged with the proper name, such as person name, location name, and date. The system uses Thing Finder created by Trouilleux [TROU1998] at Xerox, to tag the sentences. Only the sentences having tagged words, which match the question type, are carried on for answer extraction.

Answer Extraction

The answer extraction phrase tries to identify a single accurate answer from the sentences by matching the question type with each tag in each sentence. The answer

returned is based on a word whose tag is related to the question type. However, if the system generates more than one possible answer, and it cannot decide which one should be the best answer, the system will return all possible answers making it easy for the user to locate the correct answer immediately.

LASSO: A Tool for Surfing the Answer Net

LASSO was developed by Moldovan, Harabagiu et al. [MOLD1999] to obtain a correct answer to a user question expressed in a natural language. A combination of Information Retrieval and Information Extraction is used to achieve this goal. First, the system finds the semantics of query using a parser, called the question-processing module. Then, the paragraph indexing module retrieves the paragraphs that might contain the answer. Finally, the answer processing module extracts the exact answer.

Question Processing

The purpose of the Question Processing module is to define the semantics of a user's question, which include a question type, an answer type, a focus for the answer, and keywords. A user question is classified by question words, such as "what," "why," "who," "how," and "where." By looking for the type of question, an answer type can be identified. The system finds a focus of the question, which specifies what the question is about. For example, for the question, "Where is the Taj Mahal?," the question type is the word "what," the answer type is location, and the focus is the noun phrase, Taj Mahal. The process of keyword extraction is based on types of question terms, which are non-stop words, proper nouns, complex nominals, modifiers, nouns and their adjectival modifiers, verbs, and a question focus. The keywords are used to investigate paragraphs that might contain the answer.

Paragraph Indexing

According to the Boolean indexing, all keywords provided by the question-processing module are applied to retrieve documents containing the answer. To limit the set of documents retrieved, the system uses the concept of paragraph filtering. That is, only the documents containing all keywords in "n" consecutive paragraphs, where "n" is a specific integer, are selected to find the answer. Three scores are added to each paragraph. These are: a score based on a number of words from the question that are recognized in the same sequence, a score based on the number of words that separate the most distant keywords, and a score based on the number of missing keywords. Finally, the system selects a specific number of paragraphs containing the highest scores to be passed to the next module, answer processing.

Answer Processing

The Answer Processing module attempts to extract the correct answer. The system uses the help of a parser to tag semantic information, such as proper names, monetary units, and dates, to all terms of the paragraphs. Only sentences, which have the same semantic type as that of the answer type, are selected as answer candidates. To find a correct answer from the answer candidates, each answer candidate is analyzed by a scoring method depends on factors, such as the number of question words including their positions, punctuation signs, and the sequence of each answer candidate. The answer candidate having the largest score is chosen to be the most correct answer.

QALC – Question-Answering Program of the Language and Cognition Group at LIMSI-CNRS

QALC (the Question-Answering program of the Language and Cognition group at LIMSI-CNRS) system is developed by Ferret et al. [FERR2000] to find specific answers to 200 Natural language questions extracted from volumes 4 and 5 of the TREC

collection. The questions are first analyzed to find the meaningful connections between the words in the questions using linguistic relationships. The question terms are extracted as keywords with some heuristics that improve the search method. The system then indexes a set of documents to each question that might contain the answer to that question. The question/sentence pairing strategy is used to find the answer for each question. The system involves six major modules: natural language question analysis, term extraction, automatic indexing and variant conflation, named entity recognition, document ranking and thresholding, and question/sentence pairing.

Natural Language Question Analysis

Natural language question analysis is performed by a special parser based on linguistic knowledge. Ferret et al. [FERR2000] make use of TreeTagger developed by Stein and Schmid [STEI1995] to handle the syntactic and semantic categories. Each parsed question is assigned a syntactic pattern describing the structure of the question. A question type and a target that is an answer type corresponding to the question are assigned to each parsed question as well.

Term Extraction

Term extraction extracts necessary question terms from the analyzed questions. Moreover, the system tries to expand every term that has modifiers. An example given by Ferret et al. [FERR2000, p.4] is the sentence "*What is the name of the US helicopter pilot shot down?*" The following terms are extracted by the system: "US helicopter pilot," "helicopter pilot," "pilot," and "shoot." The system ignores the question word and the prepositional phrase, tries to expand the noun phrase, and uses the original form for each term. For example, the system will use the root form "shoot" for the word "shot."

Automatic Indexing and Variant Conflation

The purpose of this module is to use the question terms to retrieve documents where the specific answers might exist. The FASTR system developed by Jacquemin [JACQ1999] is employed to help the QALC system collect the documents containing the question terms. To improve the search method, the system makes use of CELEX database [CELE1998] and WordNet1.6 [FELL1998] to expand each term with variant terms having the same root morpheme and with variant terms having the same meaning, respectively.

Document Ranking

The number of documents retrieved by the document-indexing module for each question maybe large. The Document Ranking module attempts to reduce the number of the documents. First, this module ranks the documents using a weighting method. Only the 100 best-ranked documents are selected. The weighting method relies on the number of question terms found in each document, the type of the question terms (i.e., proper name, common name), the class of the question terms (i.e., original term, morphological terms, synonym terms), and the length of the terms.

Named Entity Recognition

The Named Entity Recognition module labels the terms in the documents sent from document ranking module with the named entities, such as PERSON, ORGANIZATION, and NUMBER.

Question/Sentence Pairing

For each question, the Question/Sentence Pairing module divides all the relevant documents sent from Named Entity Recognition module into sentences. Vectors of words of the question and the sentences are constructed. A weight is assigned to every pair of the question with each sentence by calculating a similarity measure between their

vectors. The similarity measure is based on the words shared by the question and the sentence, and the word features (i.e., synonym words, named entities). Finally, the sentence having the highest score is selected as the best answer.

In addition, the system attempts to find a possible answer if the method described above cannot. The system straightforwardly searches for the selected documents for that question without partitioning the documents into sentences.

This chapter reviews some of the interesting previous QA research, which leads to create the IR and AE module of WebNL system. The techniques used for each system are presented along with some examples. The next chapter provides an overview of technologies related to the IR and AE module.

CHAPTER 3
INFORMATION ON RELATED TECHNOLOGY

This chapter provides a brief introduction and background on the Extensible Markup Language (XML) including components related to this research. It also provides a brief introduction to WordNet.

Extensible Markup Language (XML)

In 1998, the World Wide Web Consortium (W3C) approved XML, the Extensible Markup Language, as a derivative of the Standard Generalized Markup Language (SGML) [W3C1998]. XML is a meta-language or language describing other languages, which allow a user to design his/her own customized markup language. The focus of the language is defining information about a document rather than the display the information. XML allows the user to place semantic tags of their own design as markups (e.g., <COURSE>) on the contents of a document. This allows XML to be an appropriate tool for describing a huge amount of information, thereby supporting knowledge representation and knowledge retrieval. In addition, XML provides an uncomplicated process to implement document types, to access its documents and retrieve their contents (e.g., by using XQL), and to share documents across the web.

The contents in a XML document is defined as a hierarchical tree pattern, containing many components including elements, attributes, and contents, using root-child-parent-sibling relationships. The structure of each XML document is based on its XML schema or its Document Type Definition (DTD). A XML document is called

"well-formed," if it has correctly nested tags. A valid document is one that conforms to a certain DTD or Schema.

To build a XML tree structure in memory, the W3C [W3C1998] offers a standard representation called the XML Document Object Model (DOM). A DOM parser is used to validate XML documents against their schema and DTD. To manipulate XML documents via DOM, an Application Programming Interface (API) is supported in many languages, such as Java and C. Moreover, W3C [W3C1998] provides other interesting components making XML more powerful. For example, the Extensible Stylesheet Language Transformation (XSLT) is used to format XML documents and transform those documents into other data formats, such as HTML. The XML Linking Language (XLink) is used to describe links between resources. The XML Pointer Language is used to point to contents of documents, and XML Query is used to access and retrieve the information stored in XML documents using query languages such as XQL and XMLQL.

The following subsections of this chapter present some components of XML, which are used in the Information Retrieve (IR) and Answer Extraction (AE) section of WebNL. Included is a summary of the Parser for XML, DOM, and XQL.

Parser for XML

XML is a meta-markup language used to represent information within a XML document. To process the XML tags, a system needs an XML parser. The parser parses the document, checks the validity of the document, and then generates either events or a data structure. XML parsers can be classified into two types: the Simple API for XML (SAX) and the Document Object Model (DOM). The former uses an event-based approach, meaning that the parser reads the text sequentially and when a start tag, end tag, attribute, or other item is found, SAX calls specific methods. The latter, DOM,

represents XML documents as a tree structure that is stored in memory. DOM provides a standard set of interfaces for manipulating contents in an XML document. Although DOM has more features than SAX, DOM has a larger memory requirement than SAX. WebNL uses DOM to parse its XML document. A summary of this model is described in the next section.

Document Object Model (DOM)

The Document Object Model (DOM) defines DOM Application Programming Interfaces (API) to dynamically navigate and manipulate the contents of XML documents. By parsing XML files, DOM pictures the XML document as a tree structure. This tree consists of nodes that are components (such as elements, attributes, and text) of the XML document. Each node is identified by a parent-child relationship. Parsing XML documents is done by a DOM parser, such as SUN Microsystem's JAXP parser [SUN2001], IBM's XML Parser for Java (XML4J) [ALPH1998], or the XML parser from Oracle's XML Developer's Kit (XDK) [ORAC2000]. To traverse and manipulate all nodes in the DOM tree, the XML DOM document is created as an instance object, first. This object depicts all properties and the methods allowing users to operate the nodes.

Currently, W3C recommendations specify DOM into 3 levels. The following brief description of DOM is stated by the W3C DOM WG [W3CD2001, p. 2]. "Level 1 allows navigation around an HTML or XML document, and manipulation of the content in that document. Level 2 extends Level 1 with a number of features: XML Namespace support, filtered views, ranges, events, etc. Level 3 is currently a Working Draft, which means that it is under active development and subject to change as we continue to refine it."

XML Query Language (XQL)

Currently, many different approaches, such as XML-QL, Lorel, YATL, and XQL, exist for querying information in XML. Robie et al. [ROBI1998] describe the meaning of structure communities related to XML query languages as follow. XML-QL, Lorel, and YALT make the same approach to querying data from semistructured data evolved from relational databases. The database community is focused on handling large databases including integrating data from heterogeneous sources, exporting views of data, and converting data into common formats used to exchange data. XQL is developed for the document community, which is focused on full-text search, queries of structured documents, integrating full-text and structured queries, and deriving multiple presentations from a single underlying document.

The structure of XQL closely follows the structure of the Extensible Stylesheet Language (XSL). XSL provides a simple format for finding elements in XML documents. For example, CISE/courses indicates finding courses elements enclosed in CISE elements. Note that XQL is more powerful than XSL. XQL can perform the basic operations, such as accessing parent/child and ancestor/descendant relationships of a hierarchy tree, the sequence of a sibling list, and the position of a sibling list. In addition, advance operations (for example, Boolean logic, filters, indexing into collections of nodes, joins allowing subtrees of documents to be combined in queries, links allowing queries to support references as well as tree structure and, searching based on text containment) are permitted by XQL as well. The result of each XQL query is a collection of XML document nodes, which can be obtained from one or more documents. WebNL makes use of the GMD-IPSI XQL engine developed by Gerald Huck [HUCK1999] to query XML documents. The GMD-IPSI XQL engine is a Java API

implementing the XQL language supporting both DOM and SAX. For a better understanding about XQL, some examples of simple queries are given in the following table.

Table 2-1. Examples of XQL queries.

XQL Query	**Meaning of query**	**Note**
CISE	To retrieve all <CISE> elements.	This query is equivalent to ./CISE
/CISE/COURSES	To retrieve all <COURSES> elements, which are children of <CISE> elements.	The first operator ("/") means the root of the document, so <CISE> element has to be the root element. The next operator ("/") indicates hierachy, which selects from immediate children of the left-side collection.
<COURSES>//<DESCRIPTION>	To retrieving all <DESCRIPTION> elements anywhere under <COURSES> element.	The operator ("//") indicates one or more hierarchy, which selects from arbitrary descendants of the left-side collection.
CISE/*/CODE='COP5555'	To retrieve all <CODE> elements having the value equal to "COP5555," which are grand-children of the <CISE> element:	The operator ("/*/) is used to selects from grand-children of the left-side collection.
//DESCRIPTION	To retrieve all <DESCRIPTION> elements anywhere in the document.	
CISE [/COURSE/@CODE = 'COP5555']	To retrieve all <CISE> elements where the value of the <CODE> attribute of <COURSE> element is equal to "COP5555" at the root of the document.	

WordNet

WordNet is an on-line lexical reference system developed by a group of psychologists and linguists led by Miller [MILL1998] at Princeton University. It is an excellent resource for Natural Language Processing (NLP), containing elements such as

an on-line dictionary and semantic concepts. WordNet contains all of the varieties of English language: nouns, verbs, adjectives, and adverbs. In the Information Retrieval (IR) and Answer Extraction (AE) module, WordNet is used for word sense generation of similarity collections. Words in WordNet are organized in synonym sets, called synsets. Each word in WordNet can be monosemous, if it has only one sense, or polysemous, if it has two or more senses. WebNL utilizes WordNet to expand each query term to improve the performance of answer searching.

To understand the concepts of a word representation in WordNet used by WebNL, an example is given. WordNet [MILL1998] defines the noun "requirement" as having three senses:

- {requirement, demand} means a required activity.
- {necessity, essential, requirement, requisite, necessary} means anything indispensable.
- {prerequisite, requirement} means something that is required in advance.

For the user request, "What is the requirements of COP5555?", the system expands the query terms, "requirements" and "COP5555" to "requirement|demand|necessity|essential|requisite|necessary|prerequisite" and "COP5555".

This chapter provides the information on related technology, which is related to the IR and AE module of the WebNL system. The next chapter examines the design of the IR and AE module.

CHAPTER 4
DESIGN OF IR and AE MODULE

This chapter discusses the design of the IR and AE (Information Retrieve and Answer Extraction) module, which is a part of the WebNL system. The goal of the IR and AE module is to try to provide a high quality answer to a user's query. The first part of this chapter describes the overall of IR and AE module. The next part provides a process description. The last part describes each process of the IR and AE module in details.

Overall of IR and AE

WebNL divides the IR and AE module into three distinct main tasks, namely, Question Analyzing, Element Indexing, and Answer Generating. Figure 4-1 depicts the overall design of the IR and AE system. The Question Analyzing task is to find the semantic of a user's request using linguistic knowledge. The system classifies a type of a question and expected answer, extracts a head noun, a main verb, and keyword terms. Using the information obtained from Question Analyzing, the Element Indexing uses the combination of Scoring method and three searching techniques: Directory Searching, Tag Element Searching, and Keyword Matching to collect the XML documents, and index tag elements containing the answer. Answer Generating uses the returned tag elements to construct queries expressed in XQL to extract an accurate answer from the XML-Knowledge Base (XML-KB). In addition, to improve the performance of answer

searching, the IR and AE module makes use of the WordNet dictionary for word sense creation of synonym sets.

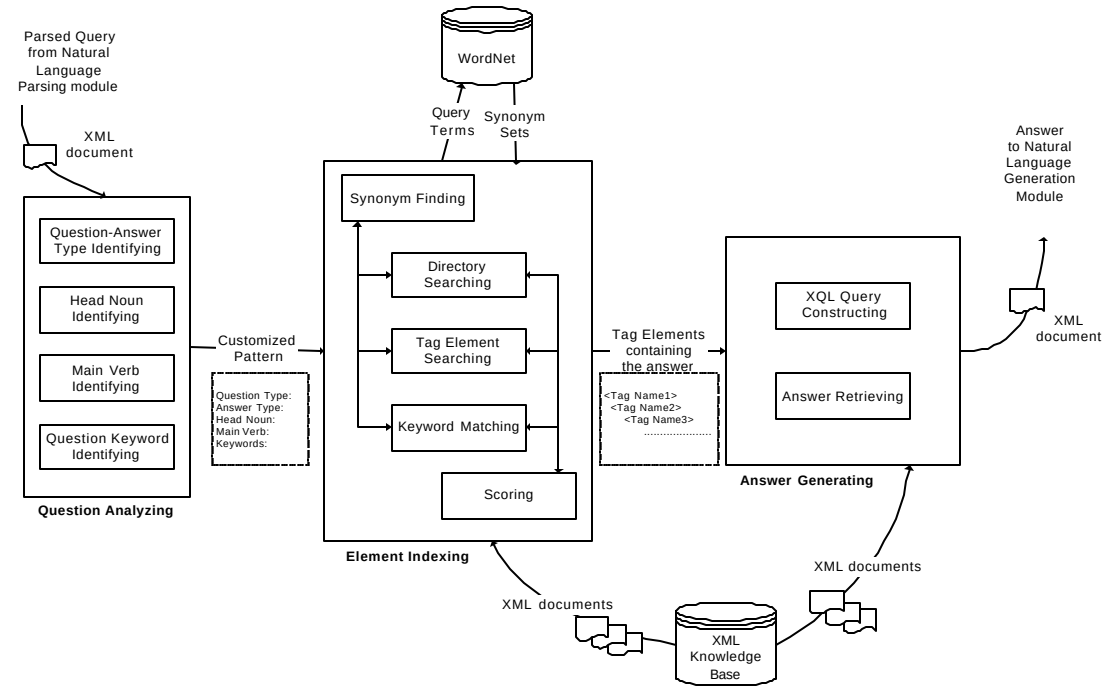

Figure 4-1. Overview of IR and AE system.

Process Description

The three tasks of the IR &AE module rely on the following processes: Question-Answer Type Identifying, Head Noun Identifying, Main Verb Identifying, Question Keyword identifying, Synonym Finding, Scoring, Directory Searching, Tag Element Searching, Keyword Matching, XQL Constructing, and Answer Generating.

Question-Answer Type Identifying uses lexical-semantic knowledge to assign a question type and an expected answer to a user request. The question-answer types usually lead the system directly to search for information requested by users.

Head Noun Identifying uses heuristics and linguistic knowledge to locate the head noun of a user request. The head noun identifies the question focus.

Main Verb Identifying, based on the heuristic search, recognizes the main verb of a user request.

Question Keyword Identifying extracts appropriate terms from a user request. Heuristics are applied to determine which terms in the query will be activated in the search expression.

Synonym Finding makes use of WordNet [MILL1998] to return a synonym set for a word. Expanding a word with its synonyms enhances the precision of the search.

Scoring uses a formula to assign a score for each search based on the type of term used (i.e., head noun, keywords) and the number of terms found in each search.

Directory Searching is the first searching technique applied to locate an answer to a user's request. This technique uses the directory file to find XML documents that probably contain the answer. Then, the system employs those returned documents to find the precise answer by navigating their tag elements. The elements containing answer are indexed to generate a query.

Tag Element Searching serves as secondary searching technique in case the Directory Searching cannot locate an answer. This technique traverses all semantic tag elements of each XML documents to index the elements containing an answer.

Keyword Matching is used as the last searching technique in case no answer is returned from the Directory Searching and the Tag Element Searching. The technique looks through all text in each XML documents in an attempt to find the term similarity of each text element to keyword terms given from the Question Keyword Identifying.

XQL Query Constructing generates a formal query using XQL. The tag elements indexed from one of the searching method (Directory Searching, Tag Element Searching,

or Keyword Matching) are used to construct an XQL query to retrieve a precise answer from the XML-KB.

Answer Retrieving utilizes the GMD-IPSI XQL Engine developed by Huck [HUCK1999] to retrieve an answer from XML-KB, and to generate the result as an XML document containing the answer. The result from Answer Generating is sent to Natural Language Generating (NLG) to process the result document, and then is returned as the answer to the user.

The following section discusses each process in details.

Question Analyzing

Question Analyzing takes the parsed user's request from the Natural Language Parsing to extract the features of the request. The main features of a request, which are used to locate a precise answer, are the type of the request and its answer, a head noun, a main verb, and keywords. The parser from Natural Language Parsing module developed by Jarosiewicz at the University of Florida constructs a structured query by substituting each word in the user's request with its corresponding linguistic concepts, such as root word, tense, and type. The structured query is represented using XML tag elements. An example illustrating the structure of a user question is shown in Figure 4-2.

The features, the type of the request and its answer, a head noun, a main verb, and keywords, can be located within the structured query. To find these features, Question Analyzing is composed of four basic processes, which are Question-Answer Type Identifying, Head Noun Identifying, Main Verb Identifying, and Question Keyword Identifying.

```xml
<?xml version='1.0'?>
<QUERY string="What is the description of COP5555?">
        <SENTENCE>
            <NOUNPHRASE>
                <PRONOUN string="what">
                    <ROOT>what</ROOT>
                    <NUMBER>indeterminate</NUMBER>
                </PRONOUN>
            </NOUNPHRASE>
            <SENTENCE string="is the description of COP5555?">
                <VERBPHRASE>
                    <VERB string="is">
                        <ROOT>be</ROOT>
                        <TENSE>present</TENSE>
                        <NUMBER>singular</NUMBER>
                    </VERB>
                    <NOUNPHRASE>
                        <NOUNPHRASE>
                            <ARTICLE string="the">
                                <ROOT>the</ROOT>
                                <TYPE>definite</TYPE>
                            </ARTICLE>
                            <NOUN string="description">
                                <ROOT>description</ROOT>
                                <NUMBER>singular</NUMBER>
                            </NOUN>
                        </NOUNPHRASE>
                        <PREPOSITIONPHRASE>
                            <PREPOSITION string="of">
                                <ROOT>of</ROOT>
                            </PREPOSITION>
                            <NOUNPHRASE>
                                <NOUN string="COP5555">
                                    <ROOT>COP5555</ROOT>
                                    <NUMBER>singular</NUMBER>
                                </NOUN>
                            </NOUNPHRASE>
                        </PREPOSITIONPHRASE>
                    </NOUNPHRASE>
                </VERBPHRASE>
            </SENTENCE>
        </SENTENCE>
</QUERY>
```


Figure 4.2. Example of a parsed question.

Question-Answer Type Identifying

The question identifier first tries to assign a category to a user's request based on the type of the question word. The corresponding answer of the request is recognized as

well. The question word of the request is extracted from the parsed request. Table 4-1 shows the question categories, which the system can cover.

Table 4-1. Question categories.

Question Categories	**Question Word**	**Answer Type**
WHO	(Who|Whom|Whose)	Person / Organization
WHERE	(Where)	Place
WHEN	(When), (What|Which) (time), (What|Which) (date)	Time
WHY	(Why)	Reason
WHATBE	(Describe|Define), (What|Who) (be) (noun phrase)	Description
WHAT	(What) (auxiliary verb)	Answer based on head noun phrase and main verb
WHATNP	(What|Which|Name) (noun phrase)	Answer type based on noun phrase after (What|Which|Name)
HOWPROCESS	How	Process
HOWADJ	(How) (adjective)	Answer type based on adjective word after (HOW)

The following describes how the system classifies the user request.

<u>WHO, WHOM, WHOSE</u>

Almost all of the focus answers of the question words “WHO,” “WHOM,” and “WHOSE” are a person, a group of people, or organizations. For example, the following question implies the person answer: “Who can recommend M.S. students to continued study toward the PhD. program?”.

However, there is an exception to the focus answers of “WHO,” “WHOM,” and “WHOSE” questions. That is these questions can seek an answer that is a description of a person rather than who this person is. For example, the question, “Who is the

supervisory committee?," has an answer that is a description of the supervisory committee.

The following rules are used when processing a "WHO," "WHOM," and "WHOSE" question.

- If the structure of the question is "(who|whom|whose) (be) Noun phrase," the system implies a description answer. "WHATBE" is assigned as the question category.

- Otherwise, the question implies the person|group of people|organization answer. "WHO" is assigned in this case as the question category.

WHERE

"WHERE" questions directly map into the answer type "Location." The system assigns the question category "WHERE" to all "Where" questions.

WHEN

The answer type given to "WHEN" questions is time. "WHEN" is assigned for all "WHEN" questions.

WHY

The answer type "reason" and the question category "WHY" are assigned to "WHY" questions. However, WebNL currently is unable to process this type of question.

DESCRIBE, DEFINE

"DESCRIBE" and "DEFINE" questions imply a description answer. "WHATBE" is assigned for question category.

WHAT, WHICH

"WHAT" and "WHICH" questions are rather confusing. The answer type for these questions is based on the focus words and the structure of the question. The

following rules are applied to assign the answer types and question categories to "WHAT" and "WHICH" questions.

- If the structure of questions is "(what|which) (be) (noun phrase)," the answer type "description" and the question category "WHATBE" are assigned.

- If the structure of questions is "(what|which) (noun phrase)," the answer type is defined by the pronoun after (what|which). The question category "WHATNP" is assigned to the question.

- Otherwise, the answer is based on the head noun, the main verb, and the keywords extracted from question. The system assigns the question category, "WHAT," to the questions.

Table 4-2. Examples of question category assignment.

Question: Question Category:	What is the description of COP5555? WHATBE
Question: Question Category:	Who is the graduate coordinator? WHATBE
Question: Question Category:	Why must I form a committee? WHY
Question: Question Category:	When should I form my supervisor committee WHEN?
Question: Question Category:	How do I form a committee? HOWPROCESS
Question: Question Category:	What materials should I submit when I apply? WHATNP
Question: Question Category:	Show me a summary of the graduate web pages. WHATBE
Question: Question Category:	How many hours can I transfer? HOWADJ

<u>HOW</u>

Two rules are applied for "HOW" questions when assigning the answer type and question category.

- If the structure of the "HOW" question is "(how) (adjective) (…)," the answer type is defined by the adjective word after the question word. "HOWADJ" is assigned as the question category to the questions.

- Otherwise, the answer type is "process" and the question category "HOWPROCESS" is assigned.

Head Noun Identifying

The head noun embedded in a user's question is used to define the question focus, which is the main information required by the question. The heuristics used to investigate a head noun are as follow:

- The first noun phrase from the user question is recognized as a head noun.
- A head noun can consist of a noun with its modifiers.
- The noun is considered as the focus noun of the user question.
- Article words and preposition words are ignored.

A head noun is extracted in the form of its root. The root form of words is necessary for answer searching, which is explained in the next section. The examples of head noun identifying are illustrated in Table 4-3.

Table 4-3. Examples of head noun identifying.

Question: Head Noun:	What is the description of COP5555? description COP5555
Question: Head Noun:	Who is the graduated coordinator? Graduate coordinator
Question: Head Noun:	Why must I form a committee? committee
Question: Head Noun:	What materials should I submit when I apply? Material

Main Verb Identifying

A main verb found in each user request represents the primary relationship between the head noun and the other noun phrases of the request. The main verb leads the system to search for a more precise answer. A main verb is found by searching for the first verb of a parsed user request.

Similar to Head Noun Identifying, Main Verb Identifying extracts the main verb from a parsed request in the form of its root. Examples of main verb identifying are shown in Table 4-4.

Table 4-4. Examples of main verb identifying.

Question: Main Verb:	What is the description of COP5555? be
Question: Main Verb:	Who is the graduated coordinator? be
Question: Main Verb:	Why must I form a committee? form
Question: Main Verb:	What materials should I submit when I apply? submit

Question Keyword Identifying

Question Keyword Identifying extracts a set of keywords, which are embedded in a user request. Keywords are used to generate the query expression for answer searching to obtain a precise answer.

The following rules are applied to select appropriate words from a parsed user request as keywords:

- Named entities, nouns, noun modifiers, and verbs are selected as keywords.
- Question words, preposition words, question marks, punctuations, and non-stop words are ignored.

Similar to head noun and main verb identifying, keywords extracted from a parsed query are in their root form. Examples of keyword identifying are given in Table 4-5.

Table 4-5. Examples of question keyword identifying.

Question: Keywords:	What is the description of COP5555? {be, description, COP5555}
Question: Keywords:	Show me a summary of the graduated web pages. {summary, graduate, web page}

Element Indexing

The aim of Element Indexing is to search for a precise answer to a user's request within the XML KB using the features of the request—a question-answer type, a head noun, and keywords. Element Indexing takes these features from Question Analyzing as its input. To retrieve an answer, the system uses three searching methods: Directory Searching, Tag Element Searching, and Keyword Matching. Both Directory Searching and Tag Element Searching try to search for a correct answer by traversing elements in the XML documents. Directory Searching employs a directory document to help the system perform a quick first search to retrieve a small number of XML documents that possibly contain the answer. Then, the system searches the retrieved documents for an answer using the head noun and keywords (called the query terms). On the other hand, Tag Element Searching searches for an answer be examining all of the content within the XML documents. Keyword Matching evaluates the degree of similarity between each XML document and the query terms. Together, the three searching methods attempt to find an accurate answer to a user's request:

- Directory Searching serves as the first attempt at answer searching.
- Tag Element Searching is used as a secondary search in case Directory Searching cannot retrieve XML documents containing an answer.
- If Tag Element Searching is unsuccessful, the system performs Keyword Matching as the last searching method.

The resulting element node holding a correct answer (generated by Element Indexing) is passed to the Answer Generating task to extract the answer.

Two additional techniques, Synonym Finding and Scoring, are employed to enhance the searching methods. The Synonym Finding technique is used to acquire a set

of synonyms of the desired words from the question using WordNet developed by Miller [MILL1998]. The Scoring technique computes a score for each search to find the most accurate answer to a user's request.

Note that to navigate and manipulate the contents of the XML documents, the system needs an XML parser. The parser parses the document, checks the validity of the document, and then generates either events or a data structure. The system utilizes the DOM parser, the Oracle XML Parser release 9.0.1 contained in Oracle's XML Developer's Kit (XDK) [ORAC2000], to parse the XML documents used in the answer searching processes.

In the remainder of this section, we discuss these approaches. First, the representation of XML knowledge base documents, which are defined in the knowledge base (called XML-KB), is briefly described. Then, the two techniques, Synonym Finding and Scoring, are discussed in details. Finally, the three searching methods (Directory Searching, Tag Element Searching, and Keyword Matching) are explained.

Representation of XML Knowledge Base Documents

Currently, fourteen XML knowledge base documents developed by Nadeau at the University of Florida exist as the knowledge base in XML-KB. These documents comprise the information in the CISE grad web pages. An XML knowledge base document is shown in Figure 4-3.

In Figure 4-3, the root element of the XML document is the <GRAD_PAGES> element. Elements under the root, which are labeled with semantic tag names, such as, "FINANCIAL," and "TUITION," present information related to their tag names. A <CW> element under each semantic element maintains a list of important keywords extracted from the information inside that semantic element. Each <CONTENT>

element presents a brief description of the semantic element that is its parent. A <TEXT> element keeps the information of its parent. A <ROOT_TEXT> element maintains almost the same content as the content in <TEXT> element related to it, but the content in the <ROOT_TEXT> element is in the form of original words (root words). The representation of <ROOT_TEXT> element is used for text search in the Keyword Matching process. The details of XML-KB representation are discussed in the thesis on the XML-KB portion of WebNL.

```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <!--  updated 09/24/01  -->
+ <!--  -->
+ <!--  -->
  <!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
  <!--  ********************** Begin FINANCIAL ************************  -->
- <GRAD_PAGES lastRevised="09/24/01">
  - <FINANCIAL>
      <CW>financial assistance</CW>
      <CONTENT>Information on available financial assistance</CONTENT>
    - <FINANCIAL_ASSISTANCE>
        <CW>financial assistance option assistantship fellowship</CW>
        <CONTENT>Financial assistance options</CONTENT>
        <TEXT>Financial assistance is available on a competitive basis in the form of fellowships and assistantships. Applications for financial
          aid for the Fall term must be received by the Department no later than February 15th. Applications for assistance should be submitted
          at the same time as the application for admission to the graduate program. While financial awards are made each semester, students
          desiring aid are encouraged to apply for Fall admission because most awards are made for that semester. Special fellowships are
          available for minority students, women, and outstanding applicants. Employment is also available on a limited basis through other
          departments and organizations, both inside and outside the University. For foreign students whose native language is not English, a
          TSE (Test of Spoken English) score is required when applying for a Teaching Assistantship. Graduate assistants are required to work
          some fraction of 40 hours per week, depending on the level of the assistantship. Students holding a fellowship or an assistantship
          must not accept other forms of employment. Reappointment to assistantships requires demonstration of good scholarship and
          satisfactory progress toward the degree. This includes maintaining a grade point average of at least 3.0 on a 4.0 scale. Fellows and
          trainees are required to devote full time to their study, but graduate assistants may register for reduced study loads according to the
          schedule in the Graduate Catalog (see the section concerning Financial Aid).</TEXT>
        <ROOT_TEXT>FINANCIAL ASSISTANCE AVAILABLE COMPETITIVE BASIS FORM FELLOWSHIP ASSISTANTSHIPS APPLICATION AID FALL TERM
          RECEIVE DEPARTMENT FEBRUARY SUBMIT ADMISSION GRADUATE PROGRAM AWARD SEMESTER STUDENT DESIRE ENCOURAGE APPLY
          SPECIAL MINORITY WOMAN OUTSTANDING APPLICANT EMPLOYMENT LIMIT THROUGH ORGANIZATION INSIDE OUTSIDE UNIVERSITY
          FOREIGN NATIVE LANGUAGE ENGLISH TSE TEST SPEAK SCORE REQUIRE TEACHING ASSISTANTSHIP ASSISTANT WORK SOME FRACTION
          HOURS DEPEND LEVEL HOLDING ACCEPT REAPPOINTMENT DEMONSTRATION GOOD SCHOLARSHIP SATISFACTORY PROGRESS TOWARD
          DEGREE INCLUDE MAINTAIN GRADE POINT AVERAGE SCALE FELLOW TRAINEE DEVOTE STUDY REGISTER REDUCE LOADS ACCORD
          SCHEDULE CATALOG SECTION</ROOT_TEXT>
      </FINANCIAL_ASSISTANCE>
    - <TUITION>
        <CW>tuition payment fee</CW>
        <CONTENT>Tuition payments</CONTENT>
        <TEXT>Teaching and research assistants receive a payment of tuition fees. Excluding these tuition fees, all assistants are required to
          pay several service fees, such as an activity fee, a health fee, an athletic fee, etc. These fees are typically less than $500/term for
          international and out-of-state students and less than $200 for in-state students. Tuition payments are provided only for those
          students on teaching and research assistantships.</TEXT>
        <ROOT_TEXT>TEACHING RESEARCH ASSISTANT RECEIVE PAYMENT TUITION FEE EXCLUDE REQUIRE PAY SEVERAL SERVICE ACTIVITY
          HEALTH ATHLETIC TYPICALLY 500/TERM INTERNATIONAL OUT-OF-STATE STUDENT IN-STATE PROVIDE ASSISTANTSHIPS</ROOT_TEXT>
      </TUITION>
    + <FINANCIAL_RESPONSIBILITY>
    </FINANCIAL>
  </GRAD_PAGES>
```

Figure 4-3. XML knowledge base document.

Synonym Finding

The IR and AE system takes advantages of the WordNet dictionary [MILL1998] to improve the performance of the answer searching. WordNet is used for word sense generation of a synonym set. Synonym concepts are an important resource for the IR and

AE system because if searching is performed using only the query terms (a head noun and keywords) extracted from a user's request, the system occasionally cannot locate the answer. Thus, it is necessary to obtain synonyms to expand the query terms.

Synonym Finding is an interface between the searching method and the WordNet application. The following algorithm is used to find synonyms of a term:

- Synonym Finding is called by the system to search for a set of synonyms for a desired word.
- Synonym Finding executes the WordNet application by passing the word as its parameter.
- WordNet processes the word returning a set of synonyms to Synonym Finding.
- Synonym Finding assigns a synonym set to the word and returns the word with its synonyms.

Scoring

The scoring method is used to compute a score assigned to each search. It takes the query terms (a head noun and keywords) and a list of words (which are compared to the query terms) as its input. The formula is used to compute a score is:

score = score_all_head_noun + score_noun_of_head_noun + (1000) *

number_of_head_noun_word_found + (10) * number_of_keyword_found

The definition of each variable is defined as follow:

- Score_all_head_noun is equal to 100000 if all terms of the head noun are found in a list of words, otherwise score_all_head_noun is equal to 0.
- Score_noun_of_head_noun is equal 40000 if a noun word embedded in head noun words is found in the list of words.
- Number_of_head_noun_word_found is the number of head noun terms found in the list of words.
- Number_of_keyword_found is the number of keyword terms found in the list of words.

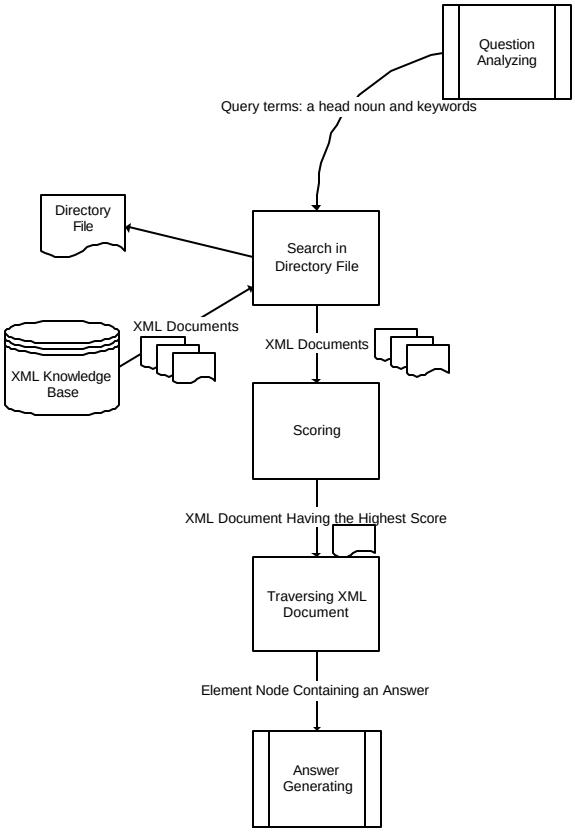

Figure 4-4. Processes for directory searching.

Directory Searching

Using the query terms (a head noun and keywords) extracted from Question Analyzing, Directory Searching is the first search method that the system applies to search for an accurate answer to a user's request. First, Directory Searching performs a search in the directory file to retrieve the XML documents probably containing an answer based on the query terms. Scoring is used to select the best XML document e.g., the one having the highest score). By traversing all of the semantic tag elements of the selected XML document, the element node containing an answer is extracted. Figure 4-4 shows the processes of Directory Searching.

The directory file and processes of Directory Searching are discussed in the following sub-sections.

Directory file

In the IR and AE system, the purpose of the directory file is to reduce the number of XML documents examined in answer searching, thereby reducing the searching time. A directory file is created as an XML document in the XML knowledge base. This file denotes all of XML documents used in the knowledge base. The description of each document is stated briefly. A part of the directory file is illustrated in Figure 4-5.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!--  updated - 9/09/01   -->
<!--  -->
<!--  -->
<!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
<!--  ********************** Begin DIRECTORY **********************  -->
<GRAD_PAGES lastRevised="09/09/01">
- <DIRECTORY domain="www.cise.ufl.edu/~ddd/grad">
  - <LISTING file="core_courses.xml">
      <CW>core course master master's degree ph.d. doctor philosophy phd ms m.s.</CW>
      <CONTENT>CISE Graduate Program core courses</CONTENT>
    </LISTING>
  - <LISTING file="overview.xml">
      <CW>overview summary</CW>
      <CONTENT>Overview of the information in the Graduate Brochure</CONTENT>
    </LISTING>
  - <LISTING file="gen_info.xml">
      <CW>general information graduate degree offer study area specialization compute computing resource</CW>
      <CONTENT>General information about the CISE graduate program</CONTENT>
    </LISTING>
  - <LISTING file="admission.xml">
      <CW>application apply admission information material mail office cise computer science department submit
        submission process</CW>
      <CONTENT>Information on admission to the CISE graduate program</CONTENT>
    </LISTING>
  - <LISTING file="financial.xml">
      <CW>financial assistance option assistantship fellowship tuition payment fee responsibility certification</CW>
      <CONTENT>Information on available financial assistance</CONTENT>
    </LISTING>
```

Figure 4-5. Part of directory file.

Each <LISTING> element is composed of a single attribute, named “file,” and two children, <CW> element and <CONTENT> element. The “file” attribute provides the XML document name used in the knowledge base. The <CW> element contains the significant keywords of the information embedded in that XML document. The <CONTENT> element provides a brief description of the related document. Currently, the WebNL knowledge base consists of 14 XML knowledge documents. The features of the WebNL knowledge base are described in the thesis on the XML Knowledge Base system.

The next section describes the searching process of the Directory Searching method.

Searching process by directory searching

Using query terms (a head noun and keywords), the system compares those terms and their relevant synonyms to each list of keywords embedded in the <CW> elements in the directory file. The scoring method is called to assign a score for each comparison. The system attempts to find a single XML document having the highest score to obtain a precise answer. The following principles are used to identify the best XML document containing an answer.

- In the directory file, the system attempts to find an XML documents whose <CW> element containing all the terms of the head noun (a noun word and all its modifiers words) and the most occurrences of the keyword terms.

- If the system cannot satisfy that first goal, it attempts to find an XML document whose <CW> element containing a head noun (consisting of a noun word and the most occurrences of its modifiers words) and the most occurrences of keywords terms.

- If neither of these approaches can decide which is the best XML document, the position of the first occurrence of the head noun in the <CW> elements is considered. The earliest position of the head noun determines which is the most important document.

If an XML document satisfies the first goal above, the document is selected as the document having the highest score. However, it is possible that more than one documents can have the highest score. This occurs when all terms of the head noun occur in the relevant <CW> element of those documents and those documents have the same number of keywords occurring in their <CW> elements. When this happens, the third goal above is applied. The system finds the position of the first occurrence of the head noun in the <CW> elements of each document. The document where the head noun

occurs earliest is selected as the best document to search for an answer. The following example illustrates selecting the best XML document using the directory file.

Example. Suppose the user request is "What are the core courses?" and the directory file is as shown in Figure 4-6.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <GRAD_PAGES lastRevised="10/08/01">
    <DIRECTORY domain="www.cise.ufl.edu/~ddd/grad">
        <LISTING file="core_courses.xml">
        <CW>core course master master's degree ph.d. doctor philosophy phd ms m.s.</CW>
          <CONTENT>CISE Graduate</CONTENT>
        <LISTING file="financial.xml">
          <CW>financial assistance option assistantship fellowship tuition payment fee responsibility certification</CW>
          <CONTENT>Information on available financial assistance</CONTENT>
        </LISTING>
        <LISTING file="masters.xml">
          <CW>master master's ms m.s. degree program admission requirement require general transfer credit supervise
                supervision supervisory committee advise advice advisement core course elective area field study specialty
                concentrate concentration thesis option nonthesis non-thesis non option exam examination progress toward</CW>
          <CONTENT>Information on the Master's Program</CONTENT>
        </LISTING>
        <LISTING file="grad_courses.xml">
          <CW>graduate course computer application design architecture engineer engineering information system program
                programming theory theoretical</CW>
          <CONTENT>The full list of available graduate courses</CONTENT>
        </LISTING>
    </DIRECTORY>
  </GRAD_PAGES>
```

Figure 4-6. Example of directory file used for the example.

According to Figure 4-6, the directory file contains four XML documents – core_courses.xml, financial.xml, masters.xml, and grad_courses.xml. Each XML document includes its own keywords embedded in the <CW> element.

The following query terms are extracted from the request.

- A head noun = “CORE COURSE” which consists of:
    - a noun : “COURSE” and
    - a modifier: “CORE”.

- A main verb = “BE”.

- Keywords = “CORE COURSE”.

To find the XML document containing the answer in the directory file, the system assigns a score to each XML document using the Scoring method. The Scoring method uses the head noun terms and their relevant synonyms, the keyword terms and their relevant synonyms, and the list of words in each <CW> element to compute a score. This score is assigned to the XML document related to the <CW> element. The following shows the elements that are compared for each document.

- For the core_courses document, the system finds the degree similarity between:
    - {CORE COURSE MASTER MASTER'S DEGREE PhD. DOCTOR PHILOSOPHY PhD MS M.S}, and {CORE | NUMCLUS | CORE GROUP | KERNEL | SUBSTANCE | CENTER | ESSENCE | GIST | HEART | INWARDNESS | MARROW | MEAT | NUB | PITH | SUM | NITTY-GRITTY | EFFECT | BURDEN}.
    - {CORE COURSE MASTER MASTER'S DEGREE PhD. DOCTOR PHILOSOPHY PhD MS M.S}, and {COURSE | COURSE OF STUDY | COURSE OF INSTRUCTION | CLASS | LINE | TREN | PATH | TRACK | ROW}.

- For the financial document, the system finds the degree similarity:
    - {FINANCIAL ASSISTANCE OPTION ASSISTANTSHIP FELLOWSHIP TUITION PAYMENT FEE RESPONSIBILITY CERTIFICATION}, and {CORE | NUMCLUS | CORE GROUP | KERNEL | SUBSTANCE | CENTER | ESSENCE | GIST | HEART | INWARDNESS | MARROW | MEAT | NUB | PITH | SUM | NITTY-GRITTY | EFFECT | BURDEN}.
    - {FINANCIAL ASSISTANCE OPTION ASSISTANTSHIP FELLOWSHIP TUITION PAYMENT FEE RESPONSIBILITY CERTIFICATION} and {COURSE | COURSE OF STUDY | COURSE OF INSTRUCTION | CLASS | LINE | TREN | PATH | TRACK | ROW}.

- For the masters document, the system finds the degree similarity between:
    - {MASTER MASTER'S MS M.S. DEGREE PROGRAM ADMISSION REQUIREMENT REQUIRE GENERAL TRANSFER CREDIT

SUPERVISE SUPERVISION SUPERVISORY COMMITTEE ADVISE ADVICE ADVISEMENT CORE COURSE ELECTIVE AREA FIELD STUDY SPECIALTY CONCENTRATE CONCENTRATION THESIS OPTION NONTHESIS NON-THESIS NON OPTION EXAM EXAMINATION PROGRESS TOWARD}, and {CORE | NUMCLUS | CORE GROUP | KERNEL | SUBSTANCE | CENTER | ESSENCE | GIST | HEART | INWARDNESS | MARROW | MEAT | NUB | PITH | SUM | NITTY-GRITTY | EFFECT | BURDEN}.

- {MASTER MASTER'S MS M.S. DEGREE PROGRAM ADMISSION REQUIREMENT REQUIRE GENERAL TRANSFER CREDIT SUPERVISE SUPERVISION SUPERVISORY COMMITTEE ADVISE ADVICE ADVISEMENT CORE COURSE ELECTIVE AREA FIELD STUDY SPECIALTY CONCENTRATE CONCENTRATION THESIS OPTION NONTHESIS NON-THESIS NON OPTION EXAM EXAMINATION PROGRESS TOWARD}, and {COURSE | COURSE OF STUDY | COURSE OF INSTRUCTION | CLASS | LINE | TREN | PATH | TRACK | ROW}.

- For the grad_courses document, the system finds the degree similarity between:
  - {GRADUATE COURSE COMPUTER APPLICATION DESIGN ARCHITECTURE ENGINEER ENGINEERING INFORMATION SYSTEM PROGRAM PROGRAMMING THEORY THEORETICAL}, and {CORE | NUMCLUS | CORE GROUP | KERNEL | SUBSTANCE | CENTER | ESSENCE | GIST | HEART | INWARDNESS | MARROW | MEAT | NUB | PITH | SUM | NITTY-GRITTY | EFFECT | BURDEN}.
  - {GRADUATE COURSE COMPUTER APPLICATION DESIGN ARCHITECTURE ENGINEER ENGINEERING INFORMATION SYSTEM PROGRAM PROGRAMMING THEORY THEORETICAL}, and {COURSE | COURSE OF STUDY | COURSE OF INSTRUCTION | CLASS | LINE | TREN | PATH | TRACK | ROW}.

The resulting score for each XML document is:

- The score of the core_courses.xml = 142,020.

- The score of the financial.xml = 0.

- The score for the masters.xml = 142,020.
- The score for the grad_courses.xml = 41,010.

Using the heuristics to identify the best XML document containing an answer, only two files, core_courses.xml and masters.xml, contain all of the head noun words (because their score are over 100000), and both have the highest score. The system

continues to find the best document by considering the position of the first occurrence of the head noun in the <CW> elements. The nead noun is found in core_courses.xml and masters.xml in positions 1 and 20, respectively. Thus, the system selects core_courses.xml as the best XML document to continue to examine for a precise answer.

Traversing an XML document

The system traverses all elements in the selected XML document in search of an answer. Figure 4-7 illustrates an algorithm for traversing the XML document to obtain the answer.

```
VisitElement(element_node, head_noun, keywords, content_search, max_score,)
{
   if (element_node has children and all query terms are not found in content_search)
   {
      content_search = content_search + text embedded in element_node's <CW>;
      get a score using head_noun, key_words, and a list of text in element_node's <CW>;
      if (all query terms are not found in content_search)
      {
         for each semantic element child of element_node
         {
            VisitElement(nd_element's child, head_noun,keywords,content_search,max_score);
         }
      }
      else
      {
         //regonize only XML document having the highest score
         if (score >= max_score)
         {
            max_score = score;
            recognize the element_node;
         }
      }
   }
   else
   {
      //regonize only XML document having the highest score
      if (score >= max_score)
      {
         max_score = score;
         recognize the element_node;
      }
   }
}
```

Figure4-7. Algorithm for traversing XML document.

The last element node recognized by the search is the element having the highest score. It is possible that one more element nodes are recognized because they all have the same highest score. The position of the head noun words found in the <CW> elements of each of the recognized nodes is considered. The recognized node where the

head noun occurs in the earliest position is selected as the best node containing the answer.

Should some of nodes have the same highest score and the same position of head noun words occurring in their <CW> element, these nodes are selected as a multiple answer. These element node(s) are sent to the next task, Answer Generating, to retrieve the answer(s) from the selected node(s) and to generate the answer document.

An example of traversing an XML document to find a correct answer is shown below.

Example

From the previous example, the user's request is "What are the core courses?" The core_courses document is selected to find a correct answer using Directory Searching. Figure 4-8 illustrates a part of core_courses document.

Using the traversing algorithm, the system visits the element node <GRAD_PAGES>, which is the root node. The root node does not have a <CW> child node. Therefore, the score given to this node is 0. The root node has one semantic element child, that is the <CORE_COURSES> element node. The system visits the <CORE_COURSES> node as a recursive call of the algorithm. The content_search variable is equal to the value in <CORE_COURSES>'s <CW>, that is "core course." The system assigns a score to the <CORE_COURSES> node by calling the Scoring method with the parameters – the value of content_search, the head noun terms (core with its synonym words and course with its synonyms), and keyword terms (core with its synonym words and course with its synonym words). This results in the <CORE_COURSES> node receiving a score of 142,020. All of the query terms are

found in the value of content_search. Therefore, the system stops searching in <CORE_COURSES>'s subchildren nodes. The score of the <CORE_COURSES> node is the maximum score at this time. Because the root node has only one child, the <CORE_COURSES> node, the algorithm stops. All content under the <CORE_COURSES> node is the generated answer. As a result, the <CORE_COURSES> node contains the exact answer to the user's request and is passed to the next task, Answer Generating, to generate the answer presented to the user.

```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <!--  updated - 9/24/01  -->
+ <!--  -->
  <!--  -->
  <!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
  <!--  ****************** Begin CORE_COURSES ******************  -->
- <GRAD_PAGES lastRevised="09/24/01">
  - <CORE_COURSES>
      <CW>core course</CW>
      <CONTENT>CISE Graduate Program core courses</CONTENT>
    - <MASTERS_CORE>
        <CW>master master's core course degree ms m.s.</CW>
        <CONTENT>The Master's Degree core courses</CONTENT>
      - <COURSE>
          <CW>course analysis algorithm cot5405</CW>
          <CONTENT>Information for Analysis of Algorithms (COT5405)</CONTENT>
          <TEXT>Analysis of Algorithms</TEXT>
        - <LINK>
            <TEXT>COT 5405</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_courses.html#COT5405</TARGET>
          </LINK>
        - <NUMBER>
            <CW>number cot5405</CW>
            <CONTENT>The course number of Analysis of Algorithms (COT5405)</CONTENT>
            <TEXT>COT 5405</TEXT>
          </NUMBER>
        - <DESCRIPTION>
            <CW>description analysis algorithm cot5405</CW>
            <CONTENT>The description of Analysis of Algorithms (COT5405)</CONTENT>
            <TEXT>This course will introduce the student to two areas. There will be a brief but intensive introduction to
              discrete mathematics followed by the study of algorithmic analysis (which comprises the bulk of the course).
              Methods for measuring complexity, order statistics. Complexity of fundamental search and sort algorithms.
              Algorithms for trees and graphs. Path problems. Graph connectivity. Dynamic programming and example
```

Figure 4-8. Part of core_courses document.

Tag Element Searching

The system performs the Tag Element Searching to find a correct answer to user's request after using the Directory Searching when no XML document is retrieved. The system accesses all XML documents in the knowledge base and tries to find the answer

by traversing all elements in each document. Similar to finding an element node containing the answer in Directory Searching, Tag Element Searching performs a traversal of an XML document using the traversing XML document algorithm. Figure 4-9 illustrates the Tag Element Searching Process.

The difference between Directory Searching and Tag Element Searching is that Directory Searching performs a search in the directory file to reduce the number of documents used to find the answer instead of searching all documents as is done in Tag Element Searching. Thus, answer searching using the Directory Searching is faster than using the Tag Element Searching.

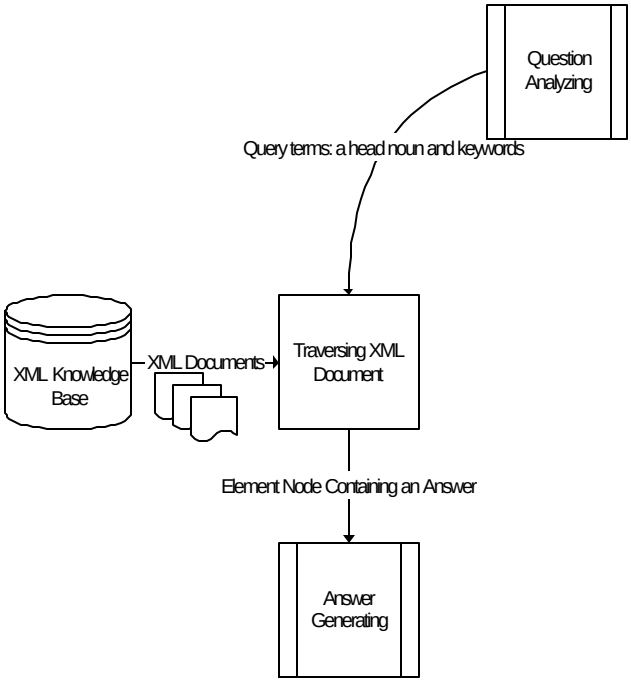

Figure 4-9. Tag element searching process.

Keyword Matching

Keyword Matching is used as the last searching method if Directory Searching and Tag Element Searching cannot extract the answer. According to Figure 4-3, all of the text is embedded in <TEXT> elements. Matching proceeds by scoring all <TEXT>

elements of the XML documents in XML-KB guided by the question-answer type and query terms extracted from Question Analyzing – a head noun and keywords. Figure 4-10 shows Keyword Matching process. An algorithm used to execute the matching process is illustrated with Figure 4-11.

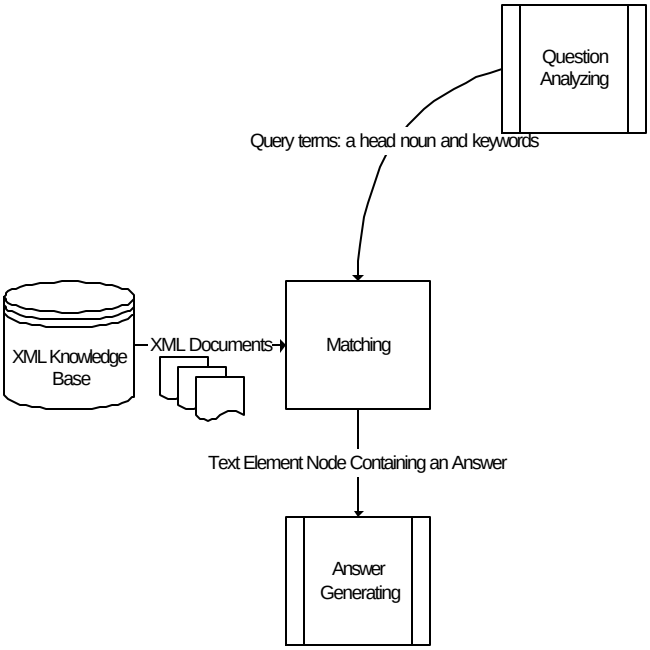

Figure 4-10. Keyword matching process.

Matching process first takes all XML documents and query terms as its input. Then, the process scores each <TEXT> element of all the XML documents guided by the head noun and the query terms. The system recognizes the parent node's <TEXT> element that has the highest score. For <TEXT> element scoring, the text content embedded in each <ROOT_TEXT> element related to that <TEXT> element is used for scoring. The content in <ROOT_TEXT> is almost identical to the content in the relevant <TEXT> element. The system generates the <ROOT_TEXT> element for each <TEXT> element using the following principle:

- The system ignores the unimportant words, such as preposition words (i.e., "in," "on," and "to), auxiliary verb words (i.e., "is," "are," and "should") and article words (i.e., "a" and "the").

- Redundant words are ignored.
- The system converts each selected word to the its original form (that is, the word's root word).

Table 4-6 shows examples of the <ROOT_TEXT> element converted from the <TEXT> element.

```
MatchingProcess(XML_documents,head_noun, keywords)
{
   for each XML_documents
   {
      max_score = 0;
      for each <ROOT_TEXT> element
      {
         get a score by call Scoring method using head_noun, key_words,
            and text content embedded in <ROOT_TEXT> element;
         if (score >= max_score && score > 0)
         {
            max_score = score;
            recognize the parent element node of this <ROOT_TEXT> element;
         }
      }
   }
}
```

Figure 4-11. Algorithm for matching process.

Table 4-6. Examples of <ROOT_TEXT> element converted from <TEXT> element.

<TEXT>Database management systems and applications, database design, database theory and implementation, database machines, distributed databases, and information retrieval</TEXT> <ROOT_TEXT>DATABASE MANAGEMENT SYSTEM APPLICATION DESIGN THEORY IMPLEMENTATION MACHINE DISTRIBUTE INFORMATION RETRIEVAL</ROOT_TEXT>
<TEXT>Several Sun 450s</TEXT> <ROOT_TEXT>SEVERAL SUN 450S</ROOT_TEXT>

According to Head Noun Identifying, Main Verb Identifying, and Keyword Identifying described in the Question Analyzing section, a head noun and keywords used as query terms are extracted in the form of their original words (i.e., the root words)

because it is easiest for the system to match those query terms to the content in the <ROOT_TEXT> element.

The element node found by Element Indexing is passed to the Answer Generating task to generate the answer.

Answer Generating

Answer Generating uses the element node containing an answer and its relevant XML document name generated by Element Indexing to create the answer in the form of an XML file. Two processes, XQL query Constructing and Answer Retrieving, generate the answer. XQL query Constructing generates a formal query using the XML Query Language (XQL). The tag elements indexed from one of searching method are used to construct an XQL query to retrieve a precise answer from the XML-KB. Answer Retrieving utilizes the GMD-IPSI XQL Engine developed by Huck [HUCK1999] to retrieve the answer and to generate the result as an XML document. The result is sent to the Natural Language Generating (NLG) system developed by Antonio at the University of Florida to process the result document and then to return the answer to the user.

```
<?xml version="1.0"?>
<q>
    <query>
        //COURSE[CW="course program programming language principle cop5555"]
    </query>
</q>
```

Figure 4-12. XQL query in form of XML file.

XQL Query Constructing

An indexed element node generated by Element Indexing is used to construct an XQL query. The constructed query is embedded in a query file as XML code. Figure 4-12 shows an XQL query in form of an XML file.

According to Figure 4-12, suppose that the <COURSE> element node indexed from Element Indexing is used to construct the query. The content in "[]" specifies the desired node. Therefore, the query, "//COURSE[CW="course program programming language principle cop5555"] ," identifies to find all <COURSE> elements that have a subelement named CW whose value is "course program programming language principle cop5555 " A constructed query file is sent to the Answer Generating process to create an answer.

Answer Retrieving

To obtain an answer, Answer Retrieving takes as input a query file and the XML document name related to a specific element node embedded in the query file. The GMD-IPSI XQL Engine acts as an interface to retrieve an answer from the XML-KB and to generate the answer written in an XML document. Figure 4-13 shows Answer Retrieving process.

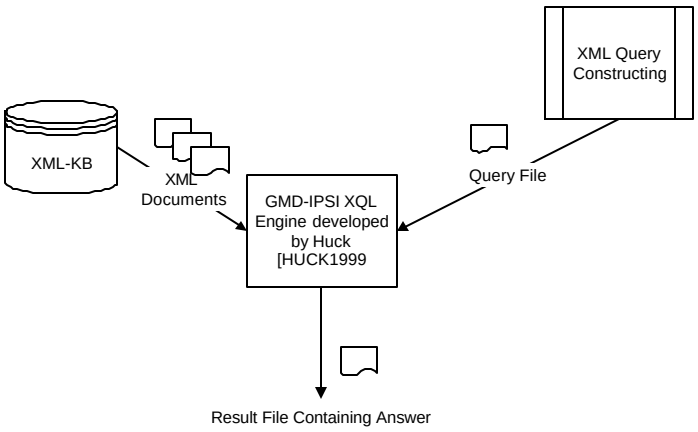

Figure 4-13. Answer retrieving process.

The result file containing the answer is sent to the Natural Language Generation module. An example of a result file is illustrated with Figure 4-14.


```
<?xml version="1.0" ?>
- <RESULT number="1">
  <QUERY string="WHAT IS THE DESCRIPTION OF COP5555?" />
  - <ANSWER type="E">
    - <COURSE>
        <CONTENT>Graduate course: Programming Language Principles</CONTENT>
        <TEXT>Programming Language Principles</TEXT>
      - <NUMBER>
          <CONTENT>Programming Language Principles course number</CONTENT>
          <TEXT>COP 5555</TEXT>
        </NUMBER>
      - <DESCRIPTION>
          <CONTENT>Description of Programming Language Principles</CONTENT>
          <TEXT>History of programming languages, formal models for specifying languages, design goals, run-time structures, and
            implementation techniques, along with survey of principal programming language paradigms.</TEXT>
        </DESCRIPTION>
      - <PREREQ>
          <CONTENT>Prerequisites for Programming Language Principles</CONTENT>
          <TEXT>COP 3530</TEXT>
        - <LINK>
            <TEXT>COP 3530</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/undergrad_pre.html#COP3530</TARGET>
          </LINK>
        </PREREQ>
      </COURSE>
    </ANSWER>
  </RESULT>
```


Figure 4-14. Example of result file.

The <RESULT> element has an attribute named “number” that identifies the number of generated answers. The user request is shown in the string attribute of the <QUERY> element. The answer is located as subelements of the <ANSWER> element. The attribute of the <ANSWER> element, “type,” identifies the accuracy of the answer. Three types of the answer are indicated by the system: “E,” “P,” and “N.” “E” means that the system extracted an accurate answer to the user’s request, “P” denotes a partial answer, and “N” identifies no answer.

In this chapter, the design of the IR and AE module is illustrated. The processes and techniques used in the module are discussed in details along with examples. The next chapter provides examples of answer extracting to user requests using the processes described in this chapter.

CHAPTER 5

EXAMPLES OF ANSWER SEARCHING TO NATURAL LANGUAGE REQUESTS

This chapter demonstrates some examples of answer searching using the techniques presented in Chapter 4. Four examples are presented to illustrate the different types of questions handled.

Example 1

This example shows the Directory Searching method for the request: "What are the PhD core classes?" First, the parser in the Natural Language Parsing module parses the request (see Figure 5-1). The parsed request is sent to the Information Retrieval and Answer Extraction (IR and AE) module to retrieve an answer from the XML knowledge base.


```
  <?xml version="1.0" ?>
- <QUERY string="What are the PH.D core classes?">
  - <SENTENCE>
    - <NOUNPHRASE>
      - <PRONOUN string="what">
          <ROOT>what</ROOT>
          <NUMBER>indeterminate</NUMBER>
        </PRONOUN>
      </NOUNPHRASE>
    - <SENTENCE string="are the PH.D core classes?">
      - <VERBPHRASE>
        - <VERB string="are">
            <ROOT>be</ROOT>
            <TENSE>present</TENSE>
            <NUMBER>plural</NUMBER>
          </VERB>
        - <NOUNPHRASE>
          - <ARTICLE string="the">
              <ROOT>the</ROOT>
              <type>definite</type>
            </ARTICLE>
          - <ADJECTIVE string="PH.D">
              <ROOT>PH.D</ROOT>
            </ADJECTIVE>
          - <ADJECTIVE string="core">
              <ROOT>core</ROOT>
            </ADJECTIVE>
          - <NOUN string="classes">
              <ROOT>class</ROOT>
              <NUMBER>plural</NUMBER>
            </NOUN>
          </NOUNPHRASE>
        </VERBPHRASE>
      </SENTENCE>
    </SENTENCE>
  </QUERY>
```


Figure 5-1. Parsed request for "What are the PhD core classes?".

In the IR and AE module, Question Analyzing analyzes the parsed request to find the semantic of the request. Table 5-1 shows the results.

Table 5-1. Features of analyzed request for “What are the PhD core classes?”.

Features	**Analyzed Value**	**Note**
Question Type	WHATBE	
Answer Type	DESCRIPTION	
Head Noun	PhD CORE CLASS	The head noun is obtained by searching for the first noun phrase of the request.
Main Verb	BE	The first verb found in the request is denoted as the main verb.
Focus Noun	CLASS	The focus noun of the question usually is the main noun of the head noun.
Keywords	{PhD, CORE , CLASS}	All terms of the request except question words, preposition words, and non-stop words are analyzed as keywords.

Element Indexing makes use of the features of the request to perform answer searching. Directory Searching is the first searching method applied. The system utilizes the directory file to retrieve a small number of documents containing the answer. Each XML document described in the directory file is assigned a score by measuring the degree of similarity between the query terms (head noun, focus noun, and keywords) and the list of significant keywords from that document. The document obtaining the highest score is selected as the document containing the answer. The score assigned to each file in the directory file is shown in Table 5-2.

According to Table 5-2, the document, core_courses.xml, is selected as having the highest score. The system examines all of the element nodes in core_courses.xml to find a node containing the answer. Using the traversing XML document algorithm, the system assigns a score to each visited node. The node receiving the highest score is indexed. The element node obtaining the highest score, <PhD_CORE>, is indexed as the

node containing the answer (see Figure 5-2). Note that a symbol, “*”, indicates the indexed element.

Table 5-2. Results from scoring each file in directory file for “What are the PhD core classes?”.

File Name	Score
core_courses.xml	143030
overview.xml	1010
gen_info.xml	41010
admission.xml	41010
financial.xml	0
masters.xml	42020
engineer.xml	0
phd.xml	42020
contacts.xml	41010
undergrad_prereqs.xml	41010
faculty.xml	1010
labs.xml	1010
grad_courses.xml	41010
undergrad_courses.xml	41010

```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <!--  updated - 9/24/01   -->
+ <!--   -->
+ <!--   -->
  <!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
  <!--  ******************* Begin CORE_COURSES *******************   -->
- <GRAD_PAGES lastRevised="09/24/01">
  - <CORE_COURSES>
      <CW>core course</CW>
      <CONTENT>CISE Graduate Program core courses</CONTENT>
    + <MASTERS_CORE>
      <!--   -->
* - <PHD_CORE>
        <CW>ph.d. doctor philosophy phd degree core course</CW>
        <CONTENT>The Ph.D. core courses</CONTENT>
        <TEXT>The Ph.D. core courses consist of all of the M.S. core courses plus COT6315.</TEXT>
        <ROOT_TEXT>PHD CORE COURSE CONSIST MS COT6315</ROOT_TEXT>
      - <COURSE>
          <CW>course analysis algorithm cot5405</CW>
          <CONTENT>Information for Analysis of Algorithms (COT5405)</CONTENT>
          <TEXT>Analysis of Algorithms</TEXT>
        - <LINK>
            <TEXT>COT 5405</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_courses.html#COT5405</TARGET>
```

Figure 5-2. Location of indexed element node for “What are the PhD core classes?”.

The indexed node is passed to Answer Generation, which constructs a XQL query and retrieves the final answer. XQL Query Constructing creates the following query :

//CORE_COURSES/PhD_CORE[CW="PhD. doctor philosophy phd degree core course"].

The XML query engine uses the query to retrieve the answer and to generate the answer file. The result file is illustrated with Figure 5-3. The answer file then is passed to the next module, Natural Language Generating, to create the natural language answer for the user.


```
<?xml version="1.0" ?>
- <RESULT number="1">
    <QUERY string="WHAT ARE THE PH.D CORE CLASSES?" />
  - <ANSWER type="E">
    - <PHD_CORE>
        <CONTENT>The Ph.D. core courses</CONTENT>
        <TEXT>The Ph.D. core courses consist of all of the M.S. core courses plus COT6315.</TEXT>
      - <COURSE>
          <CONTENT>Information for Analysis of Algorithms (COT5405)</CONTENT>
          <TEXT>Analysis of Algorithms</TEXT>
        - <LINK>
            <TEXT>COT 5405</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_courses.html#COT5405</TARGET>
          </LINK>
        - <NUMBER>
            <CONTENT>The course number of Analysis of Algorithms (COT5405)</CONTENT>
            <TEXT>COT 5405</TEXT>
          </NUMBER>
        - <DESCRIPTION>
            <CONTENT>The description of Analysis of Algorithms (COT5405)</CONTENT>
            <TEXT>This course will introduce the student to two areas. There will be a brief but intensive introduction to discrete
              mathematics followed by the study of algorithmic analysis (which comprises the bulk of the course). Methods for measuring
              complexity, order statistics. Complexity of fundamental search and sort algorithms. Algorithms for trees and graphs. Path
              problems. Graph connectivity. Dynamic programming and example applications: matrix decomposition, FFT. Theory of NP-
              Completeness.</TEXT>
          </DESCRIPTION>
        </COURSE>
            ...
            ...
            ...
      - <COURSE>
          <CONTENT>Information for Formal Languages and Computation Theory (COT6315)</CONTENT>
          <TEXT>Formal Languages and Computation Theory</TEXT>
        - <LINK>
            <TEXT>COT 6315</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_courses.html#COT6315</TARGET>
          </LINK>
        - <NUMBER>
            <CONTENT>The course number of Formal Languages and Computation Theory (COT6315)</CONTENT>
            <TEXT>COT 6315</TEXT>
          </NUMBER>
        - <DESCRIPTION>
            <CONTENT>The description of Formal Languages and Computation Theory</CONTENT>
            <TEXT>Introduction to theoretical computer science including formal languages, automata theory, Turing machines and
              computability.</TEXT>
          </DESCRIPTION>
        </COURSE>
      </PHD_CORE>
    </ANSWER>
  </RESULT>
```


Figure 5-3. Result file for "What are the PhD core courses?".

<u>Example 2</u>

This example shows an application of the Tag Element Searching method for the request: "What is the description of COP5555?" In the IR and AE module, Question Analyzing analyzes the parsed request to find the semantic of the request. Table 5-3 shows the features of the analyzed request.

Table 5-3. Features of analyzed request for "What are the description of COP5555?".

Features	**Analyzed Value**	**Note**
Question Type	WHATBE	The system ignores the word, "DESCRIPTION" for the "DESCRIPTION" answer type. Thus, the head noun and keywords contains only the word "COP5555".
Answer Type	DESCRIPTION	
Head Noun	COP5555	
Main Verb	BE	
Focus Noun	COP5555	
Keywords	{COP5555}	

First, Directory Searching attempts to find an answer. The score assigned to each file in the directory file using the Scoring method is shown in Table 5-4.

Table 5-4. Results from scoring each file in directory file for "What are the description of COP5555?".

File Name	**Score**
core_courses.xml	0
overview.xml	0
gen_info.xml	0
admission.xml	0
financial.xml	0
masters.xml	0
contacts.xml	0
engineer.xml	0
phd.xml	0
undergrad_prereqs.xml	0
faculty.xml	0
labs.xml	0
grad_courses.xml	0
undergrad_courses.xml	0

According to Table 5-4, all documents in the directory have no score, so no document is returned as an answer. Therefore, the system performs the secondary search, Tag Element Searching. Using the traversing XML Document Algorithm, the system traverses all element nodes in all XML documents in the knowledge base in an attempt to find a node containing the explicit answer. The Scoring method assigns a score to each visited node. The node obtaining the highest score is identified as the node containing an answer. For the request, "What are the description of COP5555?," the <COURSE>

element node found in the grad_courses XML document obtains the highest score. Therefore, this element node is indexed as the node containing the answer (see Figure 5-4). Note that a symbol, “*”, indicates the indexed element.

```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <!--  updated - 10/01/01   -->
+ <!--   -->
+ <!--   -->
  <!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
  <!--  *************** Begin UNDERGRAD_PREREQS ********************   -->
- <GRAD_PAGES lastRevised="10/01/01">
  - <GRAD_COURSES>
      <CW>graduate course</CW>
      <CONTENT>The full list of available graduate courses</CONTENT>
    + <APPLICATIONS>
    - <DESIGN>
        <CW>computer design architecture</CW>
        <CONTENT>Courses dealing with Computer Design and Architecture</CONTENT>
      + <COURSE>
      + <COURSE>
      + <COURSE>
      + <COURSE>
      </DESIGN>
    + <ENGINEERING>
    + <INFORMATION>
    - <PROGRAMMING>
        <CW>computer program programming</CW>
        <CONTENT>Courses dealing with Computer Programming</CONTENT>
      + <COURSE>
      + <COURSE>
  * - <COURSE>
        <CW>course program programming language principle cop5555</CW>
        <CONTENT>Graduate course: Programming Language Principles</CONTENT>
        <TEXT>Programming Language Principles</TEXT>
      - <NUMBER>
          <CW>number</CW>
          <CONTENT>Programming Language Principles course number</CONTENT>
          <TEXT>COP 5555</TEXT>
        </NUMBER>
      - <DESCRIPTION>
          <CW>description</CW>
          <CONTENT>Description of Programming Language Principles</CONTENT>
          <TEXT>History of programming languages, formal models for specifying languages, design goals, run-time structures, and
            implementation techniques, along with survey of principal programming language paradigms.</TEXT>
```

Figure 5-4. Location of indexed element node for “What is the description of COP5555?”.

This indexed node is passed to Answer Generation to construct the following XQL query:

//PROGRAMMING/COURSE[CW="course program programming language principle cop5555"].

The XML query engine uses this query to retrieve the answer from the grad_courses XML document and generates the answer file shown in Figure 5-5. Finally, the answer file is passed to the Natural Language Generating module.

```
<?xml version="1.0" ?>
- <RESULT number="1">
  <QUERY string="WHAT IS THE DESCRIPTION OF COP5555?" />
  - <ANSWER type="E">
    - <COURSE>
        <CONTENT>Graduate course: Programming Language Principles</CONTENT>
        <TEXT>Programming Language Principles</TEXT>
      - <NUMBER>
          <CONTENT>Programming Language Principles course number</CONTENT>
          <TEXT>COP 5555</TEXT>
        </NUMBER>
      - <DESCRIPTION>
          <CONTENT>Description of Programming Language Principles</CONTENT>
          <TEXT>History of programming languages, formal models for specifying languages, design goals, run-time structures, and
            implementation techniques, along with survey of principal programming language paradigms.</TEXT>
        </DESCRIPTION>
      - <PREREQ>
          <CONTENT>Prerequisites for Programming Language Principles</CONTENT>
          <TEXT>COP 3530</TEXT>
        - <LINK>
            <TEXT>COP 3530</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/undergrad_pre.html#COP3530</TARGET>
          </LINK>
        </PREREQ>
      </COURSE>
    </ANSWER>
  </RESULT>
```


Figure 5-5. Result file for "What is the description of COP5555?".

Example 3

This example shows the multiple answers found to the request: "Which materials are submitted when applying as a CISE graduate student?" Similar to the previous examples, the parsed request first is analyzed in Question Analyzing (see Table 5-5).

Table 5-5. Features of analyzed request for "Which materials are submitted when applying as a CISE graduate student?".

Features	**Analyzed Value**	**Note**
Question Type	WHATNP	
Answer Type	NP_TYPE	The answer type is based on the noun phrase, which follows the question word, "Which".
Head Noun	MATERIAL	The head noun is the first noun phase of the request included its modifiers.
Main Verb	SUBMIT	
Focus Noun	MATERIAL	
Keywords	{ MATERIAL, SUBMIT, APPLY, CISE, GRADUATE, STUDENT}.	

Directory Searching is applied to each XML document embedded in the directory file to assign a score based on the degree of the similarity between the query terms and a list of significant keywords of that document. A score assigned for each file in the directory file by using Scoring method is shown in Table 5-6.

Table 5-6. Results from scoring each file in directory file for "Which materials are submitted to apply for CISE graduated students?".

File Name	**Score**
core_courses.xml	0
overview.xml	0
gen_info.xml	10
admission.xml	141030
financial.xml	0
masters.xml	0
contacts.xml	0
engineer.xml	0
phd.xml	0
undergrad_prereqs.xml	10
faculty.xml	10
labs.xml	0
grad_courses.xml	10
undergrad_courses.xml	10

According to Table 5-6, the document, admission.xml, obtains the highest score, so it is selected as the document to examine. To locate the node containing the answers, the system traverses all element nodes in this document and assigns a score to each visited node. The node obtaining the highest score is indexed. More than one element node is indexed as the node containing the answer. See Figure 5-6. Note that a symbol, "*", indicates the indexed element.

The indexed nodes are passed to Answer Generation to construct the XQL queries shown below:

//CISE_MAIL/MATERIAL[CW="material copy application"],

//CISE_MAIL/MATERIAL[CW="material personal statement"],

//CISE_MAIL/MATERIAL[CW="material gre g.r.e. score"],

//CISE_MAIL/MATERIAL[CW="material toefl t.o.e.f.l. score"],

//CISE_MAIL/MATERIAL[CW="material transcript university"],

//CISE_MAIL/MATERIAL[CW="material tse t.s.e. score financial assistance"],

//CISE_MAIL/MATERIAL[CW="material letter reference"], and

//CISE_MAIL/MATERIAL[CW="material application financial assistance"].

Using the constructed queries, the XML query engine retrieves multiple answers from the admission document as shown in Figure 5-7.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!-- updated 09/04/01  -->
+ <!--  -->
+ <!--  -->
<!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
<!-- ********************** Begin ADMISSION ************************** -->
- <GRAD_PAGES lastRevised="09/04/01">
  - <ADMISSION>
      <CW>application apply admission submit submission process</CW>
      <CONTENT>Information on admission to the CISE graduate program</CONTENT>
    + <APPLICATION_INFO>
    + <ADMISSION_MAIL>
    - <CISE_MAIL>
        <CW>cise computer science department mail</CW>
        <CONTENT>Materials to mail to CISE department</CONTENT>
  * - <MATERIAL>
          <CW>material copy application</CW>
          <CONTENT>Copy of application</CONTENT>
          <TEXT>Copy of application (optional if applying on-line)</TEXT>
          <ROOT_TEXT>COPY APPLICATION OPTIONAL APPLY ON-LINE</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material personal statement</CW>
          <CONTENT>Personal statement</CONTENT>
          <TEXT>Personal statement</TEXT>
          <ROOT_TEXT>PERSONAL STATEMENT</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material gre g.r.e. score</CW>
          <CONTENT>GRE scores</CONTENT>
          <TEXT>Copy of GRE scores</TEXT>
          <ROOT_TEXT>COPY GRE SCORES</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material toefl t.o.e.f.l. score</CW>
          <CONTENT>TOEFL scores</CONTENT>
          <TEXT>Copy of TOEFL scores (if international student)</TEXT>
          <ROOT_TEXT>COPY TOEFL SCORES INTERNATIONAL STUDENT</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material transcript university</CW>
          <CONTENT>Official transcripts</CONTENT>
          <TEXT>Copy of transcript from each college and university attended</TEXT>
          <ROOT_TEXT>COPY TRANSCRIPT COLLEGE UNIVERSITY ATTEND</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material tse t.s.e. score financial assistance</CW>
          <CONTENT>TSE scores</CONTENT>
          <TEXT>Copy of TSE scores (if you are an international student applying for financial assistance)</TEXT>
          <ROOT_TEXT>COPY TSE SCORES INTERNATIONAL STUDENT APPLY FINANCIAL ASSISTANCE</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material letter reference</CW>
          <CONTENT>Letters of reference</CONTENT>
          <TEXT>Three (3) letters of reference</TEXT>
          <ROOT_TEXT>LETTERS REFERENCE</ROOT_TEXT>
        </MATERIAL>
  * - <MATERIAL>
          <CW>material application financial assistance</CW>
          <CONTENT>Application for financial assistance</CONTENT>
          <TEXT>Application for financial assistance (optional)</TEXT>
          <ROOT_TEXT>APPLICATION FINANCIAL ASSISTANCE OPTIONAL</ROOT_TEXT>
        </MATERIAL>
      + <ADDRESS>
      </CISE_MAIL>
    </ADMISSION>
  </GRAD_PAGES>
```

Figure 5-6. Location of indexed element node for “Which materials are submitted to apply for CISE graduated students?”.

```xml
<?xml version="1.0" ?>
- <RESULT number="0">
  <QUERY string="Which materials are submitted to apply for graduated students?" />
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>Copy of application</CONTENT>
        <TEXT>Copy of application (optional if applying on-line)</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>Personal statement</CONTENT>
        <TEXT>Personal statement</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>GRE scores</CONTENT>
        <TEXT>Copy of GRE scores</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>TOEFL scores</CONTENT>
        <TEXT>Copy of TOEFL scores (if international student)</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>Official transcripts</CONTENT>
        <TEXT>Copy of transcript from each college and university attended</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>TSE scores</CONTENT>
        <TEXT>Copy of TSE scores (if you are an international student applying for financial assistance)</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>Letters of reference</CONTENT>
        <TEXT>Three (3) letters of reference</TEXT>
      </MATERIAL>
    </ANSWER>
  - <ANSWER type="P">
    - <MATERIAL>
        <CONTENT>Application for financial assistance</CONTENT>
        <TEXT>Application for financial assistance (optional)</TEXT>
      </MATERIAL>
    </ANSWER>
  </RESULT>
```


Figure 5-7. Result file for "Which materials are submitted to apply for CISE graduated students?".

Example 4

This example shows answer searching by Keyword Matching for the request: "Can I earn a C+ in any core course?" The parsed request first is analyzed in Question Analyzing (see Table 5-7).

Table 5-7. Features of analyzed request for "Can I earn a C+ in any core course?".

Features	**Analyzed Value**	**Note**
Question Type	WHATBE	For Yes/No question, WebNL provides the answer as the information of the request.
Answer Type	DESCRIPTION	
Head Noun	C+	The head noun is the first noun phase of the request included its modifiers.
Main Verb	EARN	
Focus Noun	C+	
Keywords	{ EARN, C+, CORE, COURSE}.	

Directory Searching is first applied, generating the scores shown in Table 5-8.

Table 5-8. Results from scoring each file in directory file for "Which materials are submitted to apply for CISE graduated students?".

File Name	**Score**
core_courses.xml	2020
overview.xml	10
gen_info.xml	0
admission.xml	0
financial.xml	0
masters.xml	2020
contacts.xml	0
engineer.xml	1010
phd.xml	0
undergrad_prereqs.xml	1010
faculty.xml	10
labs.xml	1010
grad_courses.xml	1010
undergrad_courses.xml	1010

According to the formula used in the Scoring method, if the focus noun can be found in the list of keywords of a document, that document will obtain a score of at least 40000. Table 5-8 identifies that no document contains the focus noun, therefore, no document is retrieved from Directory Searching.

The system uses Tag Element Searching as the secondary search method. Similar to Directory Searching, no document is retrieved from the Tag Element Searching. The system then employs the Keyword Searching method. Each <TEXT> element of all of the XML documents is examined to find the similarity between the text content embedded in the <TEXT> element node and query terms by using the Scoring method. The <TEXT> node obtaining the highest score is indexed as the node containing the answer.

For the request, "Can I earn a C+ in any core course? ," the <TEXT> element node found in the masters XML document obtains the highest score. Therefore, this

element node is indexed as the node containing the answer (see Figure 5-8). The parent of the indexed node is passed to Answer Generating.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!--  updated - 9/24/01  -->
<!--  -->
<!--  -->
<!DOCTYPE GRAD_PAGES (View Source for full doctype...)>
<!--  ******************** Begin MASTERS ********************  -->
<GRAD_PAGES lastRevised="09/24/01">
- <MASTERS>
    <CW>master ms m.s. degree program</CW>
    <CONTENT>Information on the Master's program</CONTENT>
  + <ADMISSION_REQUIREMENTS>
  + <GENERAL_REQUIREMENTS>
  + <TRANSFER_CREDIT>
  + <SUPERVISION>
  - <MASTERS_CORE>
      <CW>master master's core course degree ms m.s.</CW>
      <CONTENT>The Master's Degree core courses</CONTENT>
    * <TEXT>The graduate core courses must be taken by every Master's student. It is important to take the core courses
        before subsequent specialty courses that use the core course material. Students are advised to not take more than
        two core courses during any single term. A minimum 3.0 grade point average must be earned in the core courses,
        with no more than one of the course grades being a C or C+. A student must retake any core course in which the
        grade earned is D+ or lower. A maximum of one core course may be repeated a single time. Descriptions of these
        courses appear in the Core Course Description.</TEXT>
      <ROOT_TEXT>GRADUATE CORE COURSE MASTER STUDENT IMPORTANT SUBSEQUENT SPECIALTY MATERIAL ADVISE
        DURING SINGLE TERM GRADE POINT AVERAGE EARN BEING C C+ RETAKE WHICH D+ LOWER REPEAT DESCRIPTION
        APPEAR</ROOT_TEXT>
    - <LINK>
        <TEXT>Core Course Description</TEXT>
        <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_core.html</TARGET>
      </LINK>
    - <COURSE>
        <CW>course analysis algorithm cot5405</CW>
        <CONTENT>Core course: Analysis of Algorithms</CONTENT>
        <TEXT>Analysis of Algorithms (COT 5405)</TEXT>
```

Figure 5-8. Location of indexed element node for “Can I earn a C+ in any core course?”.

In Answer Generating, the XQL Query Constructing process creates the query:

//MASTERS_CORE[CONTENT="The Master's Degree core courses"].

Using this query, the XML query engine retrieves the answer from the masters document. See Figure 5-9.

```
  <?xml version="1.0" ?>
- <RESULT number="1">
    <QUERY string="Can I earn C+ in any core courses?" />
  - <ANSWER type="P">
    - <MASTERS_CORE>
        <CONTENT>The Master's Degree core courses</CONTENT>
        <TEXT>The graduate core courses must be taken by every Master's student. It is important to take the core courses
          before subsequent specialty courses that use the core course material. Students are advised to not take more than
          two core courses during any single term. A minimum 3.0 grade point average must be earned in the core courses,
          with no more than one of the course grades being a C or C+. A student must retake any core course in which the
          grade earned is D+ or lower. A maximum of one core course may be repeated a single time. Descriptions of these
          courses appear in the Core Course Description.</TEXT>
      - <LINK>
          <TEXT>Core Course Description</TEXT>
          <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_core.html</TARGET>
        </LINK>
      - <COURSE>
          <CONTENT>Core course: Analysis of Algorithms</CONTENT>
          <TEXT>Analysis of Algorithms (COT 5405)</TEXT>
        - <LINK>
            <TEXT>COT 5405</TEXT>
            <TARGET>http://www.cise.ufl.edu/~ddd/grad/grad_courses.html#COT5405</TARGET>
          </LINK>
        </COURSE>
```

Figure 5-9. Part of result file for “Can I earn a C+ in any core course?”.

This chapter presents the results of query analysis in the IR and AE module. The next chapter gives the conclusions, contributions, and limitations of the research and suggestions for further studies.

CHAPTER 6
CONCLUSIONS

Searching for information on the web has attracted tremendously interest. However, the major problem with the large-scale web search engines is that they are unable to precisely retrieve the desired information of interest to the users. This results from two difficulties: the amount of information on the web is significantly increasing every day (requiring these search engines to continually update their indexes) and using a set of unordered keywords often results in a significant number of the retrieved pages that are not relevant. Question Answering (QA) systems attempt to overcome these two problems.

We have presented a QA system called WebNL that generates a high quality answer to a natural language request. This thesis addresses the retrieval of information in WebNL using an XML document in an underlying XML document knowledge base and a combination of Information Retrieval (IR) and Answer Extraction (AE) techniques. A brief introduction and background on WordNet and the Extensible Markup Language (XML) including components related to this research are described. The methodology uses three main frameworks—Question Analyzing, Element Indexing, and Answer Generating—along with two additional techniques, Synonym Finding and Scoring.

The system classified a question according to the type of answer desired to find the question's focus. Three search strategies—Directory Searching, Tag Element Indexing, and Keyword Matching—are performed with the aim of locating the answer node in WebNL's XML knowledge base based on the focus of the user's request. To

enhance the performance of these searching strategies, the system uses Synonym Finding to expand the query terms, and Scoring to weigh the accuracy of each search result based on the query terms.

Directory Searching can improve the speed of searching if the appropriate query terms are found in the directory file. The system attempts to search for the most accurate answer to a user's request by traversing all elements in an XML document. However, if an accurate answer is not found, the system attempts to find a possible answer.

Traversing elements in a XML document performed by Directory Searching and by Tag Element Indexing always give a correct answer to a user's request if the query terms exist in lists of keywords of that document. The measure of similarity between terms embedded in each text node and query terms executed by Keyword Matching usually provides a possible answer to a user's request.

Contributions

This thesis contributes to the state-of-the-art in information searching in the following four ways. First, three main frameworks—Question Analyzing, Element Indexing, and Answer Generating—are presented as a solution to extract a high quality answer to a user's request. Second, a combination of information retrieval techniques and answer extraction techniques is applied for increasing the performance of answer searching. A number of heuristics for answer searching are efficiently designed and implemented providing an appropriate search. Third, the implementation of this project, IR and AE, is purposed to merge with other components, which have been developed and are being developed by other colleagues of WebNL project in Computer and Information Science and Engineering (CISE) department at the University of Florida, to create a new

Question Answer (QA) system called WebNL for natural language request and XML knowledge base. Finally, this project is offered to use for information searching to CISE graduated web pages.

Limitation

This project is developed to retrieve a precise answer to a user's request. The current work does not completely provide the answer to all kinds of requests. For example, the system is unable to retrieve the answer for the "Why" question words. However, if the system cannot find the explicit answer, the system attempts to retrieve the most possible answer to the user. For example, for "Yes/No" questions, the system generates an answer by searching the content covering the query terms extracted from the question. The pronoun reference is not implemented in this version of the project. Further developments could be performed to increase the performances of system more powerful.

Future Studies

The concept of WebNL system is providing precise searches, which are able to find not just keywords but the best possible answer to user's requests. To achieve the goal, information retrieval techniques and answer extraction techniques are applied for the system. High Performance QA system can be built from a more techniques than the current. The further developments to enhance the performances of WebNL for next participation are suggested as follow:

- To extract a precise of answer and to support more kinds of user's requests, the query terms extracted from a parsed user's request and the content terms embedded in the XML knowledge base can be improved through a named entity (i.e., location, number, person and organization) tagging to each term.

- Multiple clauses and comparatives of a user's request could be improved through considering the semantic of them first.

- The number of expanding query terms with their synonyms could be reduced through considering the degree meaning of those synonyms.

- The pronoun reference in a user's request could be solved through using a request history keeper.

LIST OF REFERENCES

[ALPH1998] alphaWorks. (1998). XML Parser for Java. Retrieved August 30, 2001, from http://www.alphaworks.ibm.com/tech/xml4j.

[BIKE1999] D. Bikel, R. Schwartz, and R. Weischedel. An Algorithm that Learns What's in a Name. Machine Learning-Special Issue on NL Learning, vol. 34, pp. 1-3, 1999.

[CELE1998] CELEX. (1998). Consortium for Lexical Resources. Retrieved September 5, 2001, from www.ldc.upenn.edu/readme_files/celex.readme.html/.

[CHOI2000] F.Y. Choi. Advances in independent linear text segmentation. In Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL-00), pp. 26-33, 2000.

[COOP2000] R. J. Cooper and S. M. Ruger. (2000). A Simple Question Answering System. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec9/t9_proceedings.html/.

[FERR2000] Olivier Ferret, Brigitte, Gabriel Illouz et al. (2000). QALC – the Question-Answering program of the Language and Cognition group at LIMSI-CNRS. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec9/t9_proceedings.html/.

[FLYN1999] P. Flynn, T. Allen, T. Borgman et al. (1999). Frequently Asked Questions about the Extensible Markup Language. Retrieved July 10, 2001, from http://www.ucc.ie/xml/.

[HERM1997] U. Hermjakob and R.J.Mooney. Learning Parse and Translation Decisions from Examples with Rich Context. In 35th Proceedings of the Conference of the Association for Computational Linguistics (ACL), pp. 482-489, 1997.

[HOVY2000] E. Hovy, L. Gerer, M. Junk, and C. Lin. (2000). Question Answering in Webclopedia. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec9/t9_proceedings.html/.

[HUCK1999] Gerald Huck. (1999). GMD-IPSI XQL Engine. Retrieved July 12, 2001, from http://xml.darmstadt.gmd.de/xql/index.html, 1999.

[HULL1999] David A. Hull. (1999). Xerox TREC-8 Question Answering Track Report. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html/.

[JACQ1999] Christian Jacquemin. Syntagmatic and paradigmatic representations of term variation. In Proceedings of the ACL'99, University of Maryland, pp. 341-348, 1999.

[MILL1998] George A. Miller et al. (1998). WordNet: A lexical database for the English language. Retrieved July 12, 2001, from http://www.cogsci.princeton.edu/~wn/.

[MILW2000] D. Milward and J. Thomas. From Information retrieval to Information Extraction. Proceedings of the ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval. Mill Lane, Cambridge, pp. 2-3, 2000.

[MOLD1999] Dan Moldovan, Sanda Harabagiu et al. (1999). LASSO: A Tool for Surfing the Answer Net. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html/.

[ORAC2000] Oracle Technology Network. (2000). Oracle XML Developer's Kit for Java. Retrieved January 22, 2001, from http://technet.oracle.com/tech/xml/xdk_java.html.

[ROBI1998] Jonathan Robie, Joe Lapp and David Schach. (1998). XML Query Language (XQL). Retrieved August 10, 2001, from http://www.w3.org/TandS/QL/QL98/pp/xql.html.

[SUN2001] Sun Microsystems, Inc. (2001). Java^{tm} Technology and XML. Retrieved January 22, 2001, from http://java.sun.com/xml/jaxp/index.html.

[TROU1998] Francois Trouilleux. Thingfinder prototype English version 2.0. Technical report, Xerox Research Centre Europe, Grenoble, April 1998.

[VOOR1999] E. M. Voorhee. (1999). The TREC-8 Question Answering Track Report. Retrieved July 12, 2001, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html/.

[W3C1998] W3C. (1998). Extensible Markup Language (XML). Retrieved February 2, 2001, from http://www.w3.org/XML/.

[W3CD2001] the W3C DOM WG. (2001). Document Object Model FAQ. Retrieved February 2, 2001, from http://www.w3.org/DOM/faq.

[WALZ1978] David L. Waltz. An English Language Question Answering System for a Large Relational Database. Communication of the ACM, vol. 21, pp. 526-539, 1978.

[WITT1994] I.H. Witten, A. Moffat, and T.C. Bell. Managing Gigabytes: Compressing and indexing documents and images. New York, Van Nostrand Reinhold, 1994.

BIOGRAPHICAL SKETCH

Ms. Wilasini Pridaphattharakun received a BS degree in computer science from Chiangmai University in 1995. After graduation she worked as a programmer at Toyota Motor Thailand Co. Ltd. for 8 months. Then, she was systems engineer at IBM Thailand Co. Ltd. for 14 months. She moved to Zenith Comp Co. Ltd., Thailand, and worked for 22 months as a systems engineer. Having resigned from Zenith Comp Co. Ltd., she obtained an opportunity to continue her study as an M.S. student in the Department of Computer and Information Science and Engineering (CISE) at the University of Florida. Her interests include information retrieval from knowledge base and related fields, which consist of artificial intelligence, natural language processing, database, and algorithms.