Université d'Ottawa • University of Ottawa

# Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

HLAVINA, Wratko

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.C.S.

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

TAKODO: Integrating Documents with Knowledge Bases for Information Retrieval and Knowledge Management

Douglas Skuce

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

J.-P. Corriveau

K. Barker

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

# TAKODO:
# Integrating Documents with Knowledge Bases for Information Retrieval and Knowledge Management

By

Wratko Hlavina

B.Sc. University of Ottawa, 1994

A THESIS SUBMITTED TO THE

SCHOOL OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF COMPUTER SCIENCE*

School of Information Technology and Engineering (S.I.T.E.)

University of Ottawa

Ottawa, Ontario, Canada

February 2000



* The Master of Computer Science Program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

*Your file Votre référence*
*ISBN: 0-612-90080-0*
*Our file Notre référence*
*ISBN: 0-612-90080-0*

# Abstract

**Keywords**: data browsing, knowledge management, knowledge engineering, knowledge acquisition, knowledge base, information organization, information retrieval, passage retrieval, term extraction, collocation statistics, relevance ranking.

The expanding availability of on-line information sources, such as on-line documents, corpora, and the World Wide Web, introduces some challenges. Most notably, users must search through this material to find high-quality, relevant information. This is the domain of information retrieval. Alternatively, knowledge bases present information in a very compact and structured representation. Unfortunately, their creation is labor intensive.

In order to bridge the gap between the amount of structure in the information processed by information retrieval and knowledge engineering techniques, the author presents TAKODO, a tool and a framework designed for both (i) facilitating the extraction of knowledge from unstructured text, possibly to aid in the process of creating a knowledge base, and (ii) retrieving information from natural language texts and the knowledge base, using each to their mutual advantage. TAKODO integrates several existing applications, among them a question answering system, called Text Analyzer, and a frame-based knowledge management tool, called the Knowledge Organizer, with the additional support of corpus linguistic techniques.

# Acknowledgments

A few people have been of key support for the academic side during my graduate studies, and to them, I offer my sincerest thanks.

To Douglas Skuce, my supervisor, I offer special thanks for the support in my times of need. I certainly would not be here without your help.

To the LAKE group, I give my appreciation for the hard work and good conversation. In particular, I reserve special thanks for lab buddies Judy Kavanagh, John Talbot, Jamie Prpic, and Ning Huang. From you, I learned perseverance, organization skills, and the ability to get a job done. If only I could have learned sooner.

Of course, a huge, huge, huge thank you goes to my parents, Rasto and Danica, and my brother Theodore, for no other could have been of greater service and dedication than family.

I also thank Denis Batalov, a good friend with whom I shared many technical discussions somewhat related to my thesis, as well as many non-technical discussions related only to my happiness, rather than my research.

As well, I thank my other friends, by which I mean those without any direct relationship to my thesis, other than having me in common. For you, my gratitude is shown in person. Upon graduation, I shall have thrown a grand celebration for the event, where I shall tell you how important you have been.

The list of these friends includes, among others, longtime friends Alexandre, Marc, and Karen; D&D buddies Ruth, Eric, and Gordon; the "Trio of Taekwon-Girls" Julie, Alana, and Claire; hockey fan Ben; school colleagues Istvan, Mario, Elaine, Nick, Harvinder, and Pavan; Institute co-workers Daniel, Lynton, and Darlene; and several more whose influence during my graduate studies has been remote but not without importance.

To the Taekwondo group, including the "Four Black P's" (Peter, Pélé, Pierre, and Paula), the wonderful "Taekwon-Girls," as well as Roland, Kathie, John, Vinnie, Rob, Mathieu, Bernardo, and many more, I express sincere appreciation for the company, improved self-confidence, and greater well being. To all of my dance instructors at the University of Ottawa (especially to Koryn and Taryn), thank you for the encouragement and opportunity for self-expression.

Finally, I wish to credit FCAR, the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche, for financially supporting me during most of my graduate studies. Bien qu'une bourse ne peut figurer parmi ma liste d'amis, il faut dire que l'argent fut très benéfique... sans ce support financier, je n'aurais pas le luxe de la vie hors l'étude et le travail. Donc, merci de m'avoir permis de garder ma santé mentale!

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

The expanding availability of on-line information sources, such as machine-readable texts, as well as the wealth of information on the World Wide Web, introduces both interesting problems and numerous opportunities. Among the problems, users must search through a vast and increasing sea of irrelevant material to find high-quality, relevant information, satisfying some particular need. This is the domain of information retrieval.

Alternatively, knowledge bases present information in a very compact and more structured representation, making retrieval more efficient, and automated processing, such as logical inferencing, more practical [Lenat et al., 1990]. Unfortunately, their creation is labor intensive[1], often requiring the specialized services of a qualified knowledge engineer.

Hence, it is harder to find relevant and complete knowledge bases than unstructured natural language documents; the latter remains the primary method by which people communicate [Frakes & Baeza-Yates, 1992b]. In fact, most information on the World Wide Web is expressed without machine-understandable semantics [Etzioni, 1996].

In order to bridge the gap between the amount of structure in the information processed by information retrieval and knowledge engineering techniques, the author presents **TAKODO: The Way of the Text Analyzer to Knowledge Organizer**[2]. It is both a tool and a framework designed for the dual purposes of (i) facilitating the extraction of knowledge from unstructured texts, possibly to aid in the process of creating a knowledge base, and (ii) retrieving information from natural language texts and the knowledge base, using both to their mutual advantage. In the first role, the tool is targeted

[1] This problem is known as the *knowledge engineering bottleneck*, or *knowledge acquisition bottleneck* [Luger & Stubblefield, 1993].

[2] The name originates from reliance on the Text Analyzer (TA) and the Knowledge Organizer (KO). *Do*, in Japanese, means *the way of* or *art*. Taken together, TAKODO denotes the art of transferring information between the TA and the KO.

to a more specialized audience, familiar with the fundamentals of good knowledge base construction. The latter role targets the novice, whose goal is simply to answer an information need, or understand some domain.

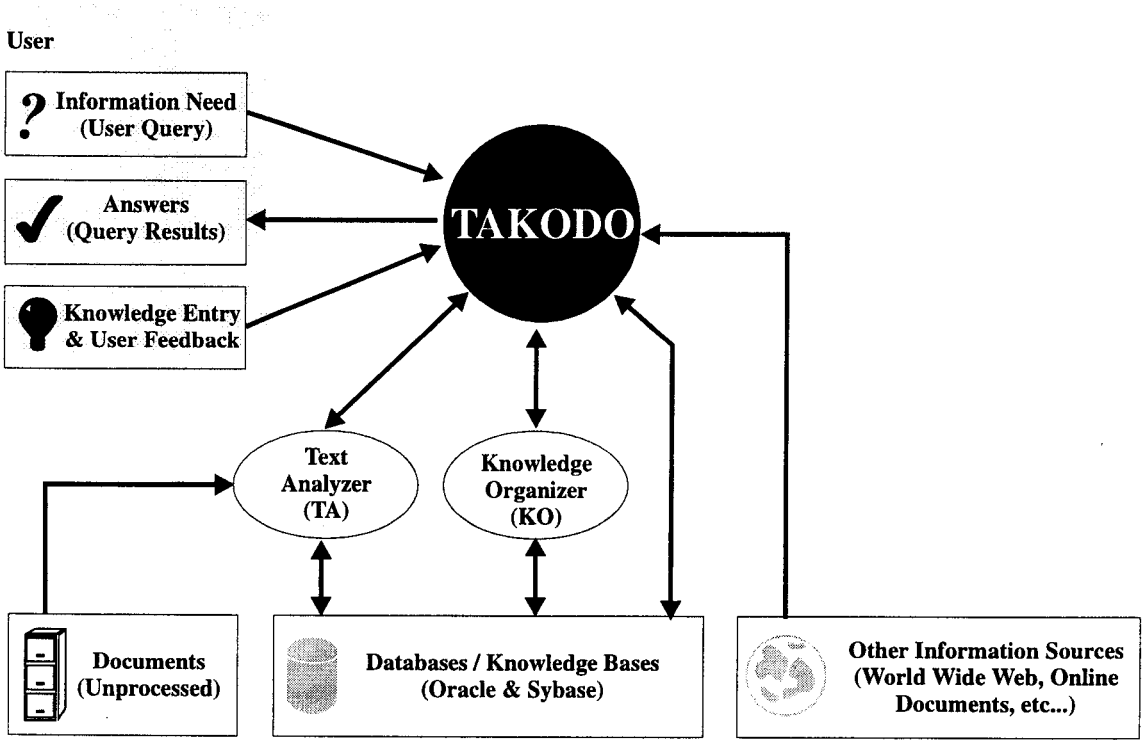



*Figure 1: Data Flow Diagram of TAKODO*

TAKODO integrates several existing applications. The first is the **Text Analyzer (TA)**, a question answering system that uses proprietary text patterns to find passages with the desired semantic meanings. The second is the **Knowledge Organizer (KO)**, a frame-based knowledge management tool with inheritance-based inferencing. Additional subsystems are needed for preprocessing, relevance ranking, and filtering. These implement various algorithms of statistics and corpus linguistics for tasks such as term extraction and collocational analysis. TAKODO manages the various sources of information and presents them through a unified interface to the user. For an overview of the system, please refer to the data flow diagram in Figure 1.

## 1.1 Goals of the Thesis

This thesis continues the work of the University of Ottawa's **Language Analysis and Knowledge Engineering group (LAKE)**, headed by Dr. Douglas Skuce. In the last decade, the group has been attempting to better integrate standard knowledge management tools and techniques with those of information retrieval. With each refinement, the group has been including more basic functionality, while reducing the complexity of the user interface.

This thesis does not attempt to further the research in information retrieval or in knowledge management, but rather, attempts to unify and apply the existing research of several areas by presenting a concrete but prototype implementation satisfying these goals, to demonstrate the feasibility of such a unifying approach.

Furthermore, the LAKE group further wishes to demonstrate the utility of the tool. In the future, the group plans on undertaking the construction of sample knowledge bases, to study and quantify the benefits.

## 1.2 Target End-User

As described, TAKODO has a dual purpose: (i) as an aid to knowledge base construction and (ii) as a search engine. More specifically, the tool's feature set accommodates the needs of users from the following categories:

- **Knowledge Engineering**: As this is among the primary focus of TAKODO, the tool includes processing that helps automate several tasks related to knowledge engineering. Beyond term extraction, it includes partial automation for the construction of various important hierarchies (hyponymy, meronymy, etc.). As well, the user interface is designed for rapid extraction of basic facts for entry into the knowledge base.

- **Domain Research**: The other primary focus of TAKODO is information retrieval. Here, the user may be, for example, a student, a researcher, or a technical support person. Two subclasses may be distinguished:

- *Users without any prior domain knowledge*: Such a user is interested in understanding a topic, and needs support for identifying and mentally organizing the important information from the available documents. For this, the knowledge management features of TAKODO provide useful navigational tools for knowledge exploration. However, TAKODO does not offer as rich navigational support in its user interface as dedicated knowledge management tools (such as the KO to which it is connected).

- *Users with some prior domain knowledge, and specific questions or information needs:* In this case, the user is better able to formulate his query. The task is less focused on navigation within the document collection, and more on information retrieval *per se*, extracting and presenting specific pieces of information. This is supported by the question answering features of the TA, and the additional information retrieval features of TAKODO.

- **Linguistics**: Lexicographers, terminologists, and translators are all interested in the words and usage patterns of existing texts, deriving their knowledge from large corpora that serve as case examples. For this, the underlying TA offers many ways to view specified word categories and word patterns in context. It is a scaleable tool, easily accommodating the large corpora required in this field. [Kavanagh, 1995]

- **Technical Writing:** A technical writer performs domain research during the writing process, in order to understand his subject and present it accurately. Hence, such a writer derives the same benefits as described above. The writer is, however, more concerned with the details of terminology, clarifying inconsistencies and ambiguities in term usage [Kavanagh, 1995].

## 1.3 Organization of the Thesis

Chapter 2 begins with a presentation of the conceptual background and discussion of relevant work related to this research, and to which references will be made in later

chapters. The topics include hypermedia, information retrieval, corpus linguistics, and knowledge engineering.

Chapter 3 continues with a presentation and evaluation of several independent systems that were integrated into TAKODO. It first introduces a legacy system, DocKMan's Knowledge Base Builder. It follows this with descriptions of the Text Analyzer and the Knowledge Organizer. All three systems are part of research projects initiated within the LAKE group.

Chapter 4 describes TAKODO itself. First, the chapter focuses on the knowledge representation and visualization aspects that guided the implementation. Then, it describes the user interface as it benefits information retrieval and knowledge management. From this, one may gain an understanding of TAKODO in use, demonstrating the steps involved in question answering and knowledge base construction.

Chapter 5 follows with the details of the implementation. The chapter traces the execution flow from initial preprocessing of source documents, all the way to the transformations required for presentation within the user interface, with emphasis on the interactive and iterative refinement of results. Along the way, there is some discussion of computational complexity issues. Then, the chapter presents the object-oriented decomposition of the system.

Finally, Chapter 6 summarizes the thesis, discusses any problems encountered, and provides insight into future work for both the theory and implementation related to TAKODO.

# Chapter 2: Related Work

This chapter presents the conceptual background that is directly related to the various subsystems of TAKODO. It includes discussions of related work and of existing systems, implementing theories covered in this chapter.

## 2.1 Hypertext and Hypermedia

**Hypertext** is text that "is not constrained to be linear," and **hypermedia** is an extension that incorporates other media such as video and audio [W3C, 1992]. Hypertext and hypermedia are represented as a network of nodes with links, often directed, associating them. Each node is a unit of information, containing an (ideally compact, complete, and cohesive) item of text or other media that may be viewed independently. The links provide navigational access from one information item to other related information items. In this respect, navigation constitutes an information retrieval process, in the broad sense.

Since it mirrors human cognition processes of associative and exploratory reasoning, the use of hypertext is easy to understand for users (unlike the sometimes complicated query syntax of classical information retrieval, to be seen later).

Note that hypertext offers little or no benefit in representing information with very definite linear structure, such as literary or expository texts, and may have limited advantages for simple hierarchically structured information. These may be presented quite adequately using the conventional paradigm of printed documents, with divisions such as chapters and headings emphasizing the subtopic structure [Pilto, 1990].

Rather, inappropriate use of hypertext can lead to very significant difficulties for the reader, due principally to the increased cognitive load of navigation and possibility of disorientation. For this reason, hypertext's practical benefit is limited to relatively small networks, unless some mechanism is employed to provide navigational cues, such as those that exist in conventional linear texts with a single narrative schema [Foltz, 1996; LinkSet, 1995].

Furthermore, authoring of hypertext presents challenges. No single and fixed organization can accommodate all needs of all readers. Given the current prevalence of hypertext with fixed topological structure, the applicability of present-day hypertext as an information retrieval tool remains limited, but research into adding "dynamic coherence" promises to allow automatic adaptation of documents and links in response to a user's personal information needs [Bell, 1995; Foltz, 1996].

## 2.2 The Internet and The World Wide Web

The Internet is a network of computer networks, connecting computers around the world. More formally, the Federal Networking Council (FNC) defines[3] it as:

> The global information system that (i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons; (ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and (iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein. [Federal Networking Council, 1995]

Examples of additional layers of information services provided by the Internet include File Transfer Protocol (FTP), Gopher, and the **World Wide Web (WWW, W3, or Web)**. Of these, the Web has gained the greater popularity, due to its ease of use and multimedia support. Originally created in 1989 by Tim Berners-Lee of the European Particle Physics Laboratory (CERN), it was described as a "wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents" [W3C, 1992]. Since then, the World Wide Web Consortium (W3C), successor to CERN for overseeing the Web's development, has broadened the definition to "the universe of network-accessible information, an embodiment of human

[3] This official definition was unanimously accepted by the FNC as of October 24, 1995.

knowledge" [W3C, 1999]. Despite this very broad definition, it is still the hypermedia aspect that, in practice, plays a pivotal role.

### 2.2.1 Components of the World Wide Web

The Web consists of various software, protocols, and conventions [W3C, 1999]. The software is modeled on a client-server architecture. The client is called a Web browser, and provides the user interface to the Web, rendering information and providing navigational support. The Web server (also called HTTP server) services information requests received from the Web browser.

The Web server and the client communicate via the **HyperText Transfer Protocol (HTTP)**, used to retrieve information in the form of Web pages, a collection of which, residing on a single host Web server, are called a Web site. These Web pages are documents expressed in **HyperText Markup Language (HTML)**, a text markup language, modeled on and specified using the Standard Generalized Markup Language (SGML). HTML specifies both the structure of information (for example, to identify paragraph boundaries or document sections and their headings) as well as its layout, with support for text containing embedded or linked media (such as images, animations, audio). Recently, HTML is being supplanted by the **eXtensible Markup Language (XML)**, which offers better support for expressing the logical structure of information, beyond the structures conventionally used in representing text documents.

Of primary importance to the Web is the method used to specify hypermedia links. The identification of abstract or physical resources is achieved via the **Uniform Resource Identifier (URI)**, which provides specific syntax to express a link as a compact string of ASCII characters [Berners-Lee et al., 1998]. A URI can be one of two kinds: a **Uniform Resource Locator (URL)**, which specifies a resource by its location (such as its host and current filename), or a **Uniform Resource Name (URN)**, which identifies resources by some persistent and globally unique identifier. At present, almost all information on the Web is located exclusively via URLs.

### 2.2.2 Growth of the Internet and World Wide Web

A premise of this research was the expanding availability of on-line information sources. This is clearly justified given the growth statistics for the Internet and, in particular, those of the Web.

Estimates of the number of Internet hosts and number of Web sites are presented in Table 1 and Table 2. Statistics reveal the Web to be doubling in size every four months [Venditto, 1998], with the total number of Web sites, as of November 1999, estimated at about 221 million [Global Reach, 1999].

| Date | Number of hosts (in Millions) | |
|---|---|---|
| | Registered hosts | Reachable hosts |
| 1993 | 1.3 | < 0.4 |
| 1994 | 2.2 | 0.6 |
| 1995 | 4.9 | 1.0 |
| 1996 | 9.5 | 1.7 |

*Table 1: Growth of the Internet [Gray, 1997]*

Furthermore, the coverage statistics [Sullivan, 1999] reveal the difficulties faced by search engines (discussed later) in providing a comprehensive index to the mass of available information: as of July 1999, there were 800 million (estimated) Web pages, but the percentage of these which were indexed by the major search engines reached from 6.3% to, at most, 31%.

| Date | Web sites | |
|---|---|---|
| | Total Number | % of Internet Traffic |
| 6/93 | 130 | 0.5 |
| 12/93 | 623 | 2.2 |
| 6/94 | 2,738 | 6.1 |
| 12/94 | 10,022 | 16.0 |
| 6/95 | 23,500 | 23.9 |
| 12/95 | 100,000 | NA |
| 6/96 | 230,000 (est.) | NA |
| 1/97 | 650,000 (est.) | NA |

*Table 2: Growth of the Web [Gray, 1997]*

## 2.3 Information Retrieval

**Information retrieval (IR)** refers to "the representation, storage, organization, and accessing of information items." [Salton, 1983] The term is often used to mean **ad-hoc retrieval**, which is covered here, and wherein the IR system manages a very large but relatively static source of information, to which users present an ad-hoc stream of queries in an interactive manner, expecting immediate results. This is opposed to other classes of IR applications, such as **categorization**, the assignment of documents arriving in a stream

and placed into predefined categories based on content, or **information filtering**, distinguished by the distribution mechanism, which is responsible for the *routing* of documents to appropriate interested parties based on user profiles describing their information needs, which are static and predefined [Smeaton & Agosti, 1994]. Although the problems are similar, the techniques used in one area are not always appropriate to the others.

Traditionally, strategies for solving retrieval needs were exemplified by libraries, with resources such as card-based catalogs and subject indexes created manually by information professionals [Kirk, 1997]. However, since the 1940's, work on the automation of retrieval led to significant developments, with the creation of many techniques, systems, and models [Salton, 1983; Frakes, 1992; Baeza-Yates & Ribeiro-Neto, 1999]. Originally, the concentration was on identifying relevant *documents* in the sense of conventional books, journal articles, and electronic texts. This is referred to as "classical information retrieval." More recently, information retrieval has been significantly influenced by the growing World Wide Web. The following discussion will briefly cover both classical information retrieval and web-based information retrieval, as well as some related fields of research.

### 2.3.1 Database Management

Information can be categorized based on the amount of usable structure in its representation. An example of highly structured information is that of the records in a relational database management system, where items sharing common features are stored in a single database table, which defines columns (or fields) for each record. Since structured information may have definite syntax and semantics, retrieval systems can be designed to provide deterministic results that completely and accurately satisfy a request for information. Software that manages such highly structured data is called a **database management system (DBMS)**. This is in contrast to information retrieval systems, which place an emphasis on unstructured information, usually in the form of ambiguous **natural language (NL)** text.

### 2.3.2 Classical Information Retrieval

Several concepts are of primary importance for an IR system; among them, there are documents, queries, and indexes.

According to Frakes [1992], "a document is a data object, usually textual, though it may also contain other types of data such as photographs, graphs, and so on." The document is the unit of information retrieval in the classical model. That is, the input to an IR system is a collection of documents, and the output is some subset of those. A document surrogate is a document (in the context of IR) that substitutes for another document. A surrogate typically contains a subset of the information in the original, for example, an abstract; surrogates are often used for efficiency, such as to reduce the amount of text that is indexed. Finally, a document base is a collection of documents gathered together, from which retrieval will be performed. When a set of documents presents some cohesion, such as being limited to a single subject domain, the document base is called a corpus.

A user query constitutes the "formal statements of information needs" [Frakes, 1992]. It describes what the user is seeking. A query language defines how a query is expressed. The expressiveness of the query language has an impact on how faithfully the information needs of the user can be represented to the IR system.

In classical IR, the kind of query considered is called a **content query**, as it specifies requirements based on the content of the retrieved information. In addition, documents may have associated structured information, such as the date of publication. This information is suitable for conventional DBMS style retrieval, and its incorporation into the query language gives rise to the term **fielded query**.

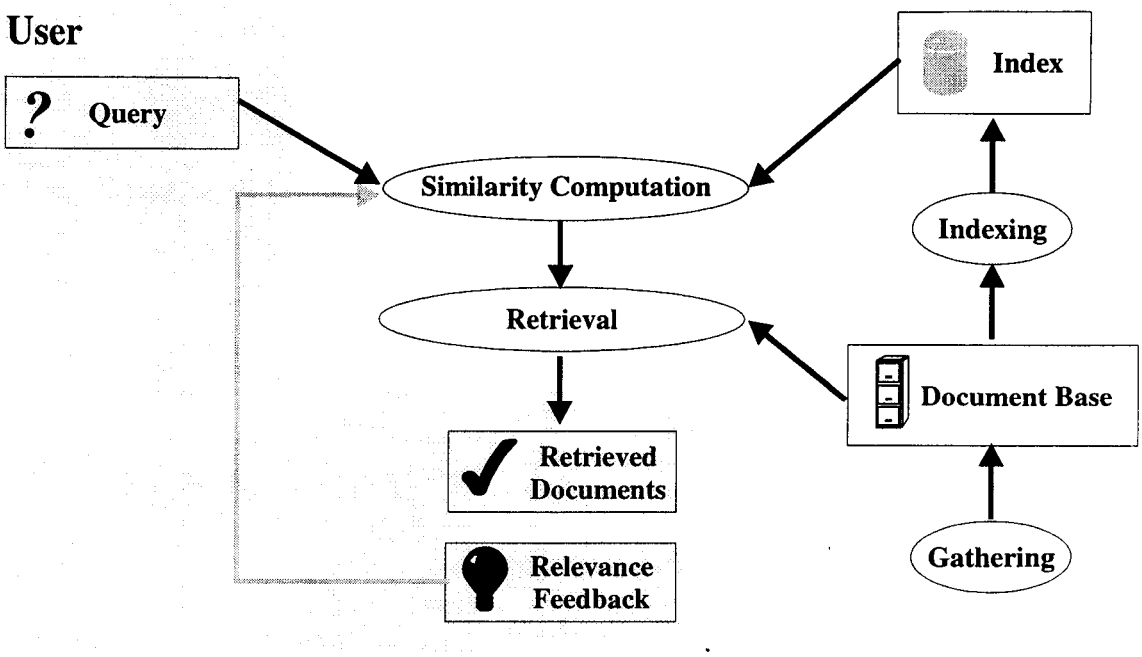



*Figure 2: The information retrieval process*

The process of retrieval (see Figure 2) can be viewed as matching the query against the documents in the document base, in order to identify those that are most relevant to the user's information needs. However, the document base is usually very large. Hence, for efficiency reasons, IR systems for ad-hoc retrieval often use an intermediate and optimized data structure, stored in an index. This index is derived from the document base. During retrieval, the index is consulted first, and from it, potentially relevant documents are identified quickly.

### 2.3.3 Web-Based Retrieval

Information retrieval in the context of the World Wide Web presents some special challenges that distinguish it from classical IR.

First of all, there is the process of gathering the documents of the document base. Unlike in classical IR, the on-line nature of the information implies that the IR system does not need to explicitly store documents, as these may be accessed at any time. However, there is the question of what to index; for this, the size and volatility of the Web prevent manual collection of documents for indexing. Instead, Web-based IR

systems must perform their own resource discovery. This is achieved by navigating through the Web, starting from some manually specified root set of documents and autonomously following links of interest [Steinberg, 1997]. Here, a document is often synonymous with a web page.

The software that performs this navigation is called a robot, or equivalently, web/search-bot, crawler, or spider [Egyhazy et al., 1997; Kirk, 1997]. Since it is responsible for generating the document base, retrieval quality will be affected by the nature of its navigation, including decisions of which links to follow and how frequently. The latter decision is important, since the robot must monitor changes to the Web, due to its distributed nature and lack of update notification.

With respect to the indexing process, the classical IR assumption of a relatively static document base allows indexing techniques that require significant computational effort. The invalidation of this assumption in Web-based retrieval imposes an additional constraint on their efficiency [Zayour, 1997].

#### *2.3.3.1 Search Engines*

Web-based IR systems that directly search the Web, as described above, are called **search engines**. An alternative is to reuse one or more existing search engines, delegating to them the retrieval task and then combining and collating their results. Such **meta-search engines** provide a single, consistent query interface to the user. Ideally, they also provide value added functionality, such as query expansion, relevance ranking, and improved result presentation and navigation abilities [Mendelson, 1997; Egyhazy et al., 1997].

#### *2.3.3.2 Navigation and Structural Queries*

Given its hypermedia nature, the Web allows information retrieval beyond the techniques of classical IR, through navigation. Often, retrieval takes on a hybrid approach: initial query results are used to begin navigation [Bell, 1995]. Thus, the constraints on the search engine can be relaxed, from finding relevant documents, to finding documents that are near the relevant information.

Furthermore, hypermedia allows another kind of query, called a structural query, which is based on the topological structure of the hypertext [Zayour, 1997]. This can be used in combination with a conventional content query, to limit the set of documents considered based on their location and access paths.

These queries may also be applied to the internal structure of Web documents, as described by their HTML or XML markup. Several proposals for a standard query language were presented for this purpose at the W3C Query Languages Workshop[4]. Among the proposals, one may mention XML Query Language (XQL) [Robie et al., 1998] and XPath [Clark & DeRose, 1999]. These permit, for example, the identification of all section headings matching some string.

### 2.3.4 Passage Retrieval

Classical IR evolved in the context of scholars and analysts seeking comprehensive bibliographical references; it does not adequately serve users seeking short answers to specific information needs [Ambroziak & Woods, 1997]. The problem lies in the retrieval of entire documents. Following retrieval, the user must spend an inordinate amount of time to manually inspect the resulting documents and identify relevant passages, hoping not to miss any. This constitutes a very significant impediment to successful and efficient retrieval, one that is doubly true when the user is seeking a short, specific fact.

Furthermore, document retrieval is especially problematic when the documents lack homogeneity. Differences in document length or structure may adversely affect how an IR system judges the relative relevance between documents. Worse still, a single document may present several different topics of discourse, only some of which are relevant to the user's information needs. Given the presence of non-relevant information, the IR system may fail to retrieve these.

Instead, passage retrieval defines the unit of retrieval to be document fragments, which are (possibly overlapping) portions of the original document. Passage retrieval is

[4] Available: http://www.w3.org/TandS/QL/QL98/

often achieved by segmenting documents prior to indexing, and then applying classical retrieval techniques to the resulting fragments, treated as individual documents [Woods, 1997].

Various strategies are used to segment a document. Arbitrary boundaries may be determined by dividing documents into equal sized pages. Although this simple strategy works effectively, boundaries motivated by additional document features may provide greater topic coherence [Hearst & Plaunt, 1993]. Hence, the divisions may attempt to follow the subtopic structure. In this case, the problem is called discourse segmentation.

Note that typical documents provide cues to their structure via conventional divisions such as chapters, paragraphs, or sentences. Additionally, the subtopic structure may be specified explicitly by section headings and subheading. When the latter are not present, the structure may be inferred using natural language processing or statistical techniques.

TextTiling [Hearst, 1994] is an algorithm for discourse segmentation in the context of IR applications that uses lexical cohesion as measured by term frequencies. The algorithm models discourse structure as a linear sequence of subtopics, generating contiguous and non-overlapping multi-paragraphs fragments.

Instead of segmenting texts prior to indexing, the IR system may directly identify relevant passages in response to user queries. Woods [1997] calls this dynamic passage retrieval, as the segmentation takes into account the broadness of a query in determining passage lengths. In [Salton et al., 1993], retrieval first applies conventional document retrieval to identify documents with relevance that satisfies a minimal threshold, and then proceeds to refine the results to smaller fragments down to the level of sentences.

### 2.3.5 Related IR Systems: Answer Extraction, Information Extraction, Question Answering

There are many fields of research that are applicable to the retrieval of information (see Table 3). Beyond passage retrieval, several of these are applicable to information needs that are very specific and expressible as a question for which the user seeks a short answer [Mollá, 1999]. For these needs, classical IR and passage retrieval

systems are evaluated solely on their ability to find documents or passages containing the answer, without much consideration for the amount of extraneous information each retrieved unit contains. In contrast, **answer extraction (AE)** implies retrieval of "those exact phrases from a set of unedited textual documents that directly answer the user query" [Mollá, 1999]. Answer extraction[5] was added to the several information retrieval research tracks addressed as part of the eighth Text REtrieval Conference[6] (TREC). The research continues in the latest conference[7], TREC-9.

All of the preceding systems return portions of documents without changes to their structure or presentation. In **information extraction (IE)**, a template for the output of answers is specified, and the task of answer extraction must fill in the slots of the template with the appropriate information [Okurowski, 1993]. The imposed structure on the output thus facilitates further machine processing, such as via conventional database management systems. Since the template has a fixed form, customized and domain-specific pattern matching techniques may be employed. The specificity of the domain (including, perhaps, a controlled vocabulary) often allows more and deeper emphasis on natural language processing (NLP). Research into information extraction was supported by the Advanced Research Projects Agency (ARPA), as part of the TIPSTER text program[8]. Research continues under the auspices of the Message Understanding Conferences[9] (MUC).

Within the field of **Artificial Intelligence (AI)**, a related problem is **question answering (QA)** [Porter et al., 1988; Keyes, 1999]. Here, information is stored in a knowledge base, and the queries are usually expressed in natural language. In response to the latter, the system performs inferencing to deduce an answer (often expressed in

---

[5] TREC uses the research heading of question answering where answer extraction may be a more appropriate term.

[6] Available: http://trec.nist.gov/ *(Note: Results of TREC-8 have not yet been published)*

[7] TREC-9, recently announced, includes seven tracks: Cross-Language Track, Filtering Track, Interactive Track, Query Track, Question Answering Track, Spoken Document Retrieval Track, and Web Track. The Filtering Track deals with information filtering, the Question Answering Track, a misnomer, deals with answer extraction, and the Web track continues the Ad-hoc Track of previous conferences, addressing ad-hoc retrieval, but in the context of the Web.

[8] Was Available: http://www.tipster.org/ *(site no longer available)*

[9] Available: http://www.muc.saic.com/

natural language). Thus, the problem relates natural language understanding with those of logical deduction and inferencing. With respect to IR, the meaning of question answering expands to include answer extraction, but with connotations of expressing queries in natural language, and possibly re-representing the output and performing some kind of inferencing. The amount of inferencing is not as deep as in AI applications, because the large document base presents computational challenges for meeting the immediate response constraints of an interactive system.

Finally, one may mention on-line help systems (such as Unix man pages or Windows WinHelp). These may reuse the research covered so far, such as text matching or hypertext, to service the information needs of users, but otherwise, their evaluation is outside the scope of this thesis.

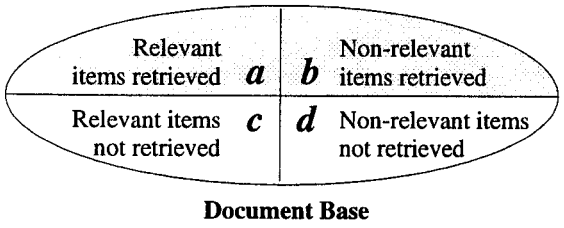
| | **Hypertext** | **Classical IR** | **Web-Based IR** | **Passage Retrieval** | **Information Extraction** | **Answer Extraction** | **Question Answering** |
|---|---|---|---|---|---|---|---|
| Source of information | Documents w/ hyperlinks | Documents | Documents w/ hyperlinks | Documents | Documents | Documents | Knowledge base (usually) |
| Volatility of information | Varies | Low | High | Varies | Varies | Varies | Low (usually) |
| Authoring, resource discovery | Manual (usually) | Manual | Automated (Robots) | Manual (usually) | Manual (usually) | Manual (usually) | Manual (usually) |
| Retrieval | Navigation | Queries (content) | Queries (content, structure) and Navigation | Queries (content) | Queries (content) | Queries (content) | Natural language queries (content) |
| Unit of retrieval | Documents | Documents | Documents (Web pages) | Document fragments | Facts (predefined templates) | Document fragments | Facts (e.g. sentences) |
| Size of information base (in practice) | Small subset of a possibly very large hyperspace | Very large | Very, very large | Very large | Varies | Varies | Small |
| Retrieval, relevance assessment | None (user performs assessment) | Statistical (word occurrence), other | Statistical (word occurrence), other | Statistical (word occurrence), other | Pattern matching, NLP | Statistical, limited NLP | Logical inferencing (usually) |
| Natural language processing, NLU | None | None to shallow | None to shallow | None to shallow | Shallow | Shallow | (Shallow to) Deep |

*Table 3: Summary of Differences between hypertext, classical IR, Web-based IR, passage retrieval, information extraction, answer extraction, and question answering*

### 2.3.6 Precision and Recall

The nature of the information handled by an IR system, specifically that of ambiguous natural language texts, and the constraints on the expressiveness of the query language, prevent retrieval from satisfying the user's information needs exactly. In order

to judge the effectiveness of retrieval, various measures have been defined. Two of the most common ones are **precision** and **recall**, defined as follows [Salton, 1989]:




$$\text{Precision} = \frac{\text{Number of relevant items retrieved}}{\text{Total number of items retrieved}} = \frac{a}{a+b}$$

$$\text{Recall} = \frac{\text{Number of relevant items retrieved}}{\text{Total number of relevant items}} = \frac{a}{a+c}$$

*Figure 3: Precision and Recall*

The goal of an IR system is to achieve the highest precision and recall, but in practice, the two are inversely related. Thus, an attempt to increase one often results in a decrease of the other, and hence a compromise must be made. Motivated by the observation that exhaustive retrieval of all relevant information is of secondary importance except in specific applications, such as patent searching, and that information needs are served as long as some relevant items are retrieved, debate exists on favoring this compromise towards greater precision [Voorhees & Harmon, 1998].

### 2.3.7 Relevance Ranking

Given that, in practice, an IR system cannot provide perfect results to match the user's information needs, it may instead present the best results first, according to some metric. This strategy is called relevance ranking. It is particularly useful for two reasons: (a) the number of retrieved items is often quite large [Frakes, 1992], therefore it allows the user to decide on how many to inspect manually; and (b) it provides relief for situations where a query might not otherwise retrieve anything, for example, when an excessively narrow query is formulated. In this case, the user may inspect the best-ranked item, and if it is not relevant, then proceed to the next, continuing until a relevant item, if it exists, is finally found.

Typically, relevance ranking strategies model documents as bags of words, and accord to each word a weight related to its frequency of occurrence statistics. Many of these strategies are minor variations of term-frequency inverse-document-frequency (tf·idf) weighting. This weighting uses the heuristic that important words will be used frequently in a document, but those used frequently in all documents have little

discriminatory power. Here, given a document base containing documents $\{D_1, D_2,..., D_N\}$, one may define the term frequency $tf_{ij}$ as the number of occurrences of a term $T_j$ in document $D_i$, and the document frequency $df_j$ of that term as the number of documents in which it occurs at least once. The tf·idf weight $w_{ij}$ of a term $T_j$ in document $D_i$ is thus defined as:

$$w_{ij} = tf_{ij} \bullet \log \frac{N}{df_j}$$

The variations on this formula address issues such as normalization of the weights according to the document length, for instance.

### 2.3.8 Relevance Feedback

Related to relevance ranking, relevance feedback is, according to Smeaton [Smeaton & Agosti, 1994], "a process whereby the relevance (or non-relevance) of a document as judged by a user is fed back to the IR system to allow the system to adjust the document ranking taking into account the attributes of known relevant documents." It provides the IR system with additional guidance on the user's information needs, which may be expressed incompletely or incorrectly due to the limits of the query language or due to the user's own lack of understanding of these needs (which may be clarified after viewing the output of an initial retrieval).

### 2.3.9 Terminological Concerns

As was mentioned for relevance ranking, the relevance of a document is often evaluated with respect to the words it contains, using, for example, occurrence statistics. Unfortunately, the words used in a query to express the information needs may not correspond exactly to those used in the document base. Hence, various techniques are employed in IR to address these terminological difficulties. The basic processes perform substitutions for words occurring as indexing terms, in documents, and in queries.

#### *2.3.9.1 Word Conflation and Disambiguation: Stemming, Lemmatization*

First, consider word conflation, in which two words are considered equal for indexing or retrieval purposes. In order to eliminate distinctions between words with similar meanings, a dictionary, if one is available, may be consulted to identify synonyms

and the root form of a word. Alternatively, stemming algorithms exist to remove common affixes. In the simplest case, handling of plural versus singular forms is often desirable.

Word conflation is a way of improving recall. However, automatic word conflation may introduce equivalencies between words that do not share the same meaning, thus decreasing the precision of the query results.

A related problem is word sense disambiguation, where a word has many possible meanings (and different root forms); the problem is to identify which one is intended in the context where the word is used. Its interest is increased precision of query results, although recall may be adversely affected when the incorrect senses are inferred by the IR system. Applications of word sense disambiguation in IR have had limited success [Kilgarriff, 1997a], with precision benefits less than 2%. A hypothesis is that queries with multiple terms inherently provide a kind of sense disambiguation; therefore, the techniques are redundant except for very short queries.

The concept of a word that includes all of its inflected forms is called a **lemma**, and the process of grouping words under their lemma is called lemmatization [Barnbrook, 1996]. Furthermore, let us define a **lexeme** as "a lexical unit taken in one sense only", and a **vocable** as the "family of lexemes sharing identical signifiers and related to each other through a common semantic component" [Meyer & Steele, 1990]. For example, the lemma *retrieval* includes the inflected forms *retrieval*, *retrieve*, and *retrieving* (in all of their senses). The vocable *retrieve (verb)* includes, among others, the two lexemes *retrieve (in the sense of rescue)* and *retrieve (in the sense of remembering)*. The previous vocable is distinct from the vocable *retrieve (noun)*, which includes, among others, the lexeme *retrieve (in the sense of a successful return of a ball, as in tennis)*.

Thus, conflation by replacing occurrences of words with their root forms or stems is an attempt at lemmatization. Rather than deriving lemmas, the preceding process can be coupled with word sense disambiguation, in which case the output is lexemes.

#### *2.3.9.2 Truncation*

The purpose of truncation in the context of IR is similar to that of word conflation, but the user requests its use explicitly at query formulation time. With truncation, a set of terms is matched based on the presence of a given prefix. For example, a query for `run*` (where * is a wildcard character) should match occurrences of *running, runner,* and any other terms beginning with same first three characters. Truncation offers more user control than stemming or dictionary-based root form lookup, for example, but note that (i) the prefixes may match words with unrelated meanings, such as *runt*, and (ii) some words exhibit minor spelling variations in their various conjugated forms, or when suffixes are added. Thus, retrieval will fail to find *ran*, and a search for `sky*` will not find the plural, *skies*.

#### *2.3.9.3 Query Expansion and Thesaural Expansion*

Query expansion is the augmenting of the terms in a query with additional terms. As with truncation, its purpose is similar to that of word conflation, and it is applied at query formulation time, rather than at indexing time.

Typically, query expansion implies thesaural expansion of the constituent terms of the query. The International Organization for Standardization (ISO) defines ten standard types of thesaural relationships, which Table 4 summarizes. These may be applied during thesaural expansion.

| Thesaural Relationship | Abbreviation | Description |
|---|---|---|
| Synonym | SYN | Same semantic meaning |
| Preferred Term | PT | Synonym that is preferred in usage |
| Related Term | RT | Associated term that is likely relevant |
| Top Term | TT | Most general term, according to the generic hierarchical relationship |
| Broader Term | BT | More general term, by some relationship |
| Narrower Term | NT | More specific term, by some relationship |
| Broader Generic Term | BTG | More general term (*hypernym*), according to the generic hierarchical relationship ("is-a") |
| Narrower Generic Term | NTG | More specific term (*hyponym*), according to the generic hierarchical relationship ("is-a") |
| Broader Partitive Term | BTP | More general term (*holonym*), according to the partitive hierarchical relationship (whole/container) |
| Narrower Partitive Term | NTP | More specific term (*meronym*), according to the partitive hierarchical relationship ("part-of") |

*Table 4: Standard ISO2788 Thesaural Relationships [ISO, 1986]*

The benefits of automatic query expansion are sometimes debated, since the losses in precision may outweigh the recall benefits, but there have been many good success stories, particularly those combining query expansion with relevance feedback [Carpinetti et al., 1998]. Given the problems, particularly those of thesaural expansion, an interactive approach is suggested, allowing the user to guide the query reformulation process, such as in the IRIS system [Yang et al., 1998].

#### *2.3.9.4 Controlled Vocabularies*

When the documents are domain-specific, the terminological difficulties may be limited through the use of a controlled vocabulary. This controlled vocabulary provides a consistent way of naming concepts. It may be used to guide information professionals during manual assignment of indexing terms, or during full-text retrieval to restrict indexing exclusively to those terms in the vocabulary. Unfortunately, because creation and use of a controlled vocabulary is domain specific, often labor intensive, and requires user familiarity during query formulation, its use has found limited applicability outside of specialized areas, such as in industry for technical documentation.

### 2.3.10 Classifications of Information Retrieval Systems

Frakes describes a faceted classification of IR systems, based on their conceptual models, file structures, query operations, and other characteristics [Frakes, 1992]. In what follows, one finds a sampling of these and other classifications [Baeza-Yates & Ribeiro-Neto, 1999].

#### *2.3.10.1 Full-Text Retrieval*

Retrieval may be performed based on manually determined indexing terms, also called keywords, which are assigned to each document. However, this assignment of indexing terms requires considerable time and skill, and its quality affects the retrieval performance. Instead, retrieval may depend directly on the content of each document, using the set of words it contains (or suitable derivations thereof) as the indexing terms. This is called full-text or free-text retrieval [Lesk, 1995]. Note that, in practice, the set of words used is not exhaustive, due to efficiency concerns during indexing.

As some concepts cannot be adequately expressed by a single word, one desirable feature in full-text systems is phrase retrieval. By phrase, one means a sequence of multiple words used in order. Phrase retrieval is especially useful when the meaning of a phrase is unrelated to the meanings of its constituent parts. For example, it would be unfortunate to search for "World Wide Web," only to find numerous and extraneous items pertaining to large arachnids living in foreign countries!

#### *2.3.10.2 Text Pattern Matching*

In the text pattern matching conceptual model, the query is a text pattern, such as a fixed string or a Unix style regular expression. During retrieval, the IR system identifies those documents containing the text pattern. Often, the results are presented with the location of the matches highlighted within the retrieved documents

Usually, the retrieval is accomplished without an index, via a full-text sequential scan of the document base. Although techniques for efficient text scanning exist, the performance remains, at best, linear with respect to the document length. Nevertheless, given current processing speeds, full text scanning remains a viable strategy for many applications.

In particular, hybrid systems that merge sequential scanning with some degree of pre-filtering or partial indexing, such as Glimpse [Manber & Wu, 1994], demonstrate the practical effectiveness. Using default indexing options, Glimpse divides the document base into 256 approximately equal sized blocks, which may cross document boundaries. The blocks are indexed using an inverted index that records occurrences with a granularity at the block level, as will be discussed later. During retrieval, the inverted index is itself searched using full-text sequential scanning, to yield a set of possibly relevant blocks. The document fragments within these blocks are subsequently scanned to verify their relevance. Glimpse's design accommodates sophisticated pattern matching with extensions to Unix style regular expressions, and further adds Boolean operators, as in the following retrieval model.

#### *2.3.10.3 Boolean Retrieval Model*

In the Boolean IR conceptual model [Salton, 1989], retrieval of documents occurs based on the presence or absence of associated indexing terms. The model derives its name from the query language, which specifies the user's requirements as Boolean conjunctions (and), disjunctions (or), and negations (not) of terms. Although this model is among the earliest and most common, it presents several limitations: it does not incorporate any relevance ranking, presuming all retrieved items of equal importance, nor does it offer any control over the number of items retrieved. Furthermore, the model is very sensitive to the query formulation, significantly more than other models: overly narrow queries return *no* items, and insufficiently narrow queries return an *excessive* quantity of items.

Extensions to the basic Boolean conceptual model do provide support for relevance ranking. One strategy, quorum-level searches, defines a sequence of queries with decreasing relevance derived from the user's original query. These proceed from very narrow formulations, consisting of conjunctions of all terms, to progressively broader formulations, consisting of disjunctions.

Another approach applies fuzzy set extensions that make use of term weighing information. In this case, the indexing terms for each document are given a weight

describing their importance to the document, as will be seen for the vector space model below. The relevance of a document with respect to a given query will then be a function of the document's term weights, for those terms used in the query. For complete details, please refer to [Salton, 1989].

In addition to the preceding operators, Boolean full-text IR systems may allow proximity constraints that specify a given term must occur within some maximum distance of another term's occurrence, perhaps with ordering restrictions. This allows the user to impose a context on word usage within the documents. Such a feature, if present, may subsume phrase retrieval.

#### *2.3.10.4 Vector Space Model*

In the vector space conceptual model, each distinct index term in the document base forms one orthogonal axis of a multidimensional vector space. In this space, each document is viewed as a vector extending from the origin to a point defined by the document's term weights along each axis. Similarly, the query is represented as a vector in this same space. The documents are then relevance ranked using a similarity measure such as the vector dot product or the cosine coefficient.

In the simple case, the value along each axis is either 0 or 1, based on the absence or presence, respectively, of the axis' term in the document or query. More frequently, the value is based on tf·idf weights.

#### *2.3.10.5 Knowledge-Based Models*

In many cases, information needs are related to ideas rather than the language used to express them. Thus, a criticism of many information retrieval models is their modeling of documents as sets of words, without any consideration for the concepts expressed in those documents, nor of the relationships between these concepts. The addition of such information to the retrieval process results in what is called knowledge-based retrieval or intelligent information retrieval.

The addition may be as simple as using a thesaurus to augment the set of terms used in a query, to achieve a form of concept recognition (see query expansion, as

discussed previously). Alternatively, knowledge-based conceptual models of information retrieval attempt to directly address the stated criticism by modeling documents as knowledge bases; their implementations often represent documents explicitly as networks of connected concept nodes, wherein retrieval is achieved via inferencing to identify those documents with high probability of relevance. An example of such an inferencing strategy is spreading activation [Salton, 1988; Crestani, 1995], wherein each node is initialized to some degree of activation, and subsequently these levels of activation are propagated iteratively throughout the network, based on the strength of the links. Upon a specified termination condition, the relevance of nodes is judged based on the final level of activation.

In practice, knowledge-based models face the challenge of extracting the knowledge from the documents. In many cases, existing systems use co-occurrence statistics that once again model documents as sets of words. Other techniques using thesauri or natural language processing (such as is used for information extraction) are highly domain dependent [Yang, 1997].

#### *2.3.10.6 Other Models*

Other models exist, but their description is outside the scope of this thesis. Instead, one may briefly mention the following:

- Probabilistic model [Salton, 1989]: Rather than consider the similarity of a document to the query, as in the vector space model, the probabilistic model predicts the probability of relevance given the relevance of prior documents. The model makes statistical assumptions to evaluate the probabilities of relevance from term occurrence information.
- Logic model [van Rijsbergen, 1986, cited in Yang, 1997; Gloor, 1997]: Herein, the documents are represented using formal logic, with logical inferencing used for retrieval. However, the model once again faces the difficulty of deriving such a representation from natural language texts using automated techniques.
- Cluster model [Gloor, 1997]: Rather than present documents in a (possibly ranked) list, the cluster model identifies similarity between documents, using

techniques of the vector space model. The documents are then arranged as cohesive clusters of self-similar documents.

### 2.3.11 Indexing

At least two meanings of indexing are relevant in IR: one is "the process of constructing document surrogates by assigning identifiers to text items" [Salton, 1989], and the other is "the building of a data structure that will allow quick searching of the text" [Baeza-Yates, 1992]. The former deals specifically with identifying index terms, whereas the latter is broader, encompassing both techniques that depend on such assigned index terms as well as those that do not. Unless explicitly stated, all references to indexing hereafter shall imply the broader definition.

For the discussion that follows, let us assume that documents are to be indexed (in the narrow sense) through automated procedures by their content, rather than by manual intervention. Specifically, let us address the issue of full-text indexing.

#### *2.3.11.1 Filtering and Automated Assignment of Index Terms*

Prior to indexing, the documents of the document base may be filtered with goals of reducing the amount of data that is indexed or of standardizing its representation, which will later facilitate retrieval [Brooks, 1997]. Typically, preprocessing includes removal of special non-alphabetic characters, normalization of spacing, capitalization, punctuation, numbers, dates, and other such transformations. It is at this stage that one considers the terminological concerns expressed previously, employing word conflation and disambiguation strategies.

Certain words appear with very high frequency and are present in most natural language texts; hence, they carry very little discriminatory value for retrieval. Examples include closed-class or function words, that is, words whose semantic meanings are fixed and which serve a structural role in language, such as determiners (*the*, *a*, *an*) or prepositions (*on, in, around*). Other cases include content words that may be nearly omnipresent when a document base consists of a single domain-specific corpus. Such words are often included in a stoplist, which lists any words that should be ignored for purposes of retrieval, and removed from use as an indexing term.

Similarly, very rare and thus low-frequency words are sometimes removed, as they are considered insignificant with respect to describing the content of documents.

These considerations of word frequency as related to their significance, or "resolving power," are based on research originating with H. P. Luhn and experiments using mathematical models of word occurrence patterns, such as Zipf's Law [cited in van Rijsbergen, 1979].

As it turns out, a large percentage of the words used in a given document have limited resolving power. Since the number of indexed word occurrences and total count of distinct indexing terms significantly affects the space and runtime efficiency of most indexing techniques, appropriate use of a stoplist provides a useful tradeoff between retrieval abilities[10] and performance.

#### *2.3.11.2 File Structures*

The efficiency concerns of IR systems are commonly addressed by lexicographical indexes, clustered file structures, or hashing-based indexes [Harman et al., 1992]. The choice of file structures impacts the space overhead, retrieval abilities, and computational complexity for index updates and retrieval. For a comparative analysis, the reader should consult some of the voluminous literature on data structures as well as descriptions of IR techniques in [Baeza-Yates & Ribeiro-Neto, 1999; Frakes & Baeza-Yates, 1992a; Witten et al., 1999; Zayour, 1997] and others.

Typically, the file structures allow quick identification of those documents associated with a given indexing term. For full-text indexing, the result of such a lookup may further specify the locations of all term occurrences, to some greater level of granularity such as the paragraph, sentence, or exact word within each document. Although increased granularity adversely affects the index size, it may provide quicker retrieval or more advanced features such as proximity constraints between occurrences.

---

[10] Thus, in full-text retrieval, any word that is removed from consideration as an indexing term cannot be used in a query for retrieval (unless the systems falls back to an expensive linear full-text scan of the document base, for example)

### *2.3.11.3 Inverted Index*

An inverted index is a lexicographic index that contains, for each indexing term of the document base, a list of pointers to all of occurrences of the term. It is a popular type of index, particularly for Boolean IR systems. Since an inverted index provides as output the set of documents containing an indexing term, a natural translation of Boolean queries emerges, replacing Boolean conjunctions, disjunction, and negations of terms in the query with intersections, unions, and differences, respectively, of document sets.

The standard implementation of an inverted index uses a main index file and a postings file (see Figure 4). The main index file stores the indexing terms in lexicographical order, often with summary statistics such as term weights. Typical data structures include sorted arrays and balanced multi-way trees such as B-trees. The posting file contains, for each distinct index term, the lists of pointers to its occurrences within the originating documents. It is essentially a linked list whose entry point is accessed from the index file.

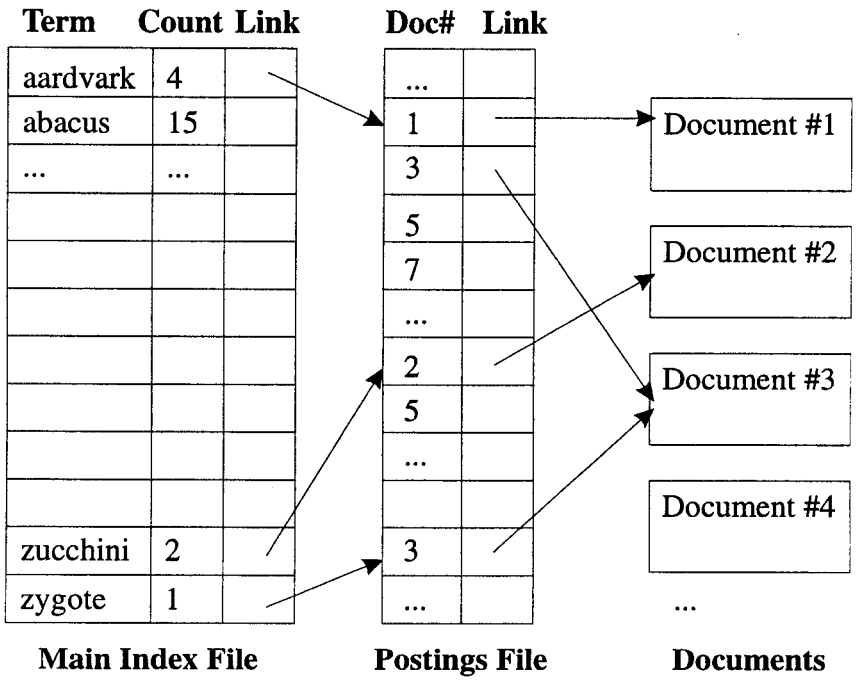



*Figure 4: Inverted index (typical implementation using a sorted array)*

Whether the index file is an array or a B-tree, search time remains logarithmic, that is, O(log $n$), where $n$ is the number of index terms. The main consideration is that updating a sorted array is far more expensive.

#### *2.3.11.4 Other Indexes*

A bitmap is simply a Boolean matrix with one axis for documents and another for index terms, where each bit is set if and only if the corresponding document contains the incident index term. The array can be viewed as a list of bit vectors for each document, identifying its index terms. With their high space overhead, bitmaps have limited applications.

Signature files are similar, but employ hashing to generate length-limited bit vectors for each index term. The signature of a document is the bitwise logical OR of all bit vectors corresponding to its set of index terms. Retrieval generates a filtered list of candidate documents. This (hopefully small) list must then be scanned to verify the presence of the indexing terms in the candidate documents.

Hash tables also employ hashing, but instead, their model is an inverted index, without the lexicographical ordering.

##### *2.3.11.4.1 PAT Structures: PAT Trees and PAT Arrays*

A PAT structure [Gonnet & Baeza-Yates, 1992] is a lexicographic index that allows rapid access to all matching *sistrings* within the document base. A sistring is the semi-infinite string that begins at some position within a document and extends to its end (it is the "tail" of a document). PAT trees index these sistrings via Patricia trees, which are digital binary tries without single-descendent nodes. PAT arrays are a different implementation, with performance tradeoffs in the interest of decreased space overhead and improved efficiency during construction.

Although implementation details are tricky, the advantage of a PAT structure lies in retrieval flexibility, allowing more sophisticated query languages. It permits, for example, efficient searching for regular expressions (including prefixes/truncation and

phrase retrieval), matches with proximity constraints, lexicographical ranges, approximate matches, longest matches, and most significant (relevance ranked) matches.

#### *2.3.11.5 Latent Semantic Indexing*

One of the difficulties involved with full-text indexing lies in the large number of indexing terms. For example, this may adversely affect the index size, such as is commonly used for Boolean retrieval, and the computational complexity for naïve implementations of solutions using the vector space conceptual model.

Instead, Latent Semantic Indexing (LSI) uses singular value decomposition (SVD) to derive a small number of indexing terms that are aggregates of the original indexing terms [Berry et al., 1994]. Unfortunately, this transformation is itself computationally expensive. Current research is ongoing to address this issue [O'Brien, 1994]. Similar techniques, using low-rank orthogonal decompositions such ULV or URV, rather than SVD, may offer computational advantages [Berry, 1996].

#### *2.3.11.6 Conceptual Indexing*

Rather than employ query expansion to alleviate differences in terminology by augmenting the list of query terms, a system may identify the concepts associated with documents during indexing. In effect, LSI is a form of conceptual indexing. Other techniques, such as those of [Woods, 1997], employ natural language processing and statistical techniques to derive a conceptual taxonomy from the source documents. This taxonomy is based on "most specific subsumer" relationships, and is stored in a knowledge base. This system indexes each document under all of its associated concepts. Woods noted that the system exhibited space and runtime efficiency concerns.

### 2.3.12 Information Retrieval Systems

At present, information retrieval is a hot topic, especially with regard to Web search engines. Rather than attempt an exhaustive[11] evaluation that would be quickly outdated, the following shall present a sampling of current and representative systems.

---

[11] At the time of this writing, a useful resource for search engine comparisons is Search Engine Watch, available online at http://www.searchenginewatch.com/.

#### *2.3.12.1 Major Web Search Engines: AltaVista, Excite, etc.*

Almost all Web search engines use full-text indexing of the contents of Web pages and plain text documents, with very little linguistic processing or knowledge-based enhancements. Perhaps the exceptions are Excite[12] and InfoSeek Ultra[13], which claim to perform conceptual indexing[14].

AltaVista[15] is among the largest[16] Web search engines. As is typical, AltaVista presents the user with a choice between natural language queries (as in a question answering system) and an advanced, Boolean query language (including a proximity operator). Experimentation suggests[17] that a natural language query is interpreted as a simple term list, without any natural language processing beyond using a stoplist to remove words like *what, who, is*. The system supports phrase retrieval and term truncation, but does not employ stemming or any other conflation techniques during indexing. Limited support for fields allows querying results by date, title, and URL, among others.

Retrieval is quick: no more than a few seconds. Web pages are relevance ranked, using term occurrence statistics, and links to the top results (about 20) are presented in order via a Web page. The latter displays the title of each retrieved link, along with a short summary (as specified by HTML META tags, or as generated by truncating the page to the first paragraph or so). Navigation allows the user to proceed to the next 20 most relevant links.

---

[12] Available: http://www.excite.com/

[13] Available: http://ultra.infoseek.com/

[14] Both search engines use technology that is proprietary; no implementation details are publicly available. The patent pending conceptual indexing technology of Excite is called Intelligent Concept Extraction (ICE). Marketing documentation [http://www.excite.com/ice/tech.html] compares it to Latent Semantic Indexing in effectiveness, but notes significantly greater computational efficiency.

[15] Available: http://www.altavista.com/

[16] As of December 1999, the number of indexed web pages was reported in excess of 250 million Web pages. Note that comparisons between search engines are sometimes contradictory and may contain marketing exaggerations.

[17] Details are not easily available. Indexing technology is a closely guarded corporate secret for many commercial search engine vendors.

Marketing documentation[18] suggests that AltaVista's indexing technology uses an inverted index, but with space optimizations that reduce its overhead to between 10% and 30%. This is despite the unusual design choice of indexing *all words and numbers*, without *any* stoplist during indexing.

Of note during relevance ranking, the major search engines often employ several heuristics, such as giving preference to query terms occurring in titles, headings, or near the start of the page. In the case of multiple search terms, those whose matches appear close together may also be given preference. Some, such as Lycos[19], rank documents by popularity, via statistics on the number of times a page was cited in other pages.

Many search engines provide some support for query reformulation via relevance feedback, by including for each retrieved item a link to find similar items. Innovations in search engine technology, as utilized in Northern Lights[20], include the automatic clustering of documents into a taxonomy of subject classifications, which is automatically derived using word co-occurrence statistics.

For resource discovery, search engines provide an interface so that Web site administrators may submit URLs for navigation and indexing by the search engine's robot.

#### *2.3.12.2 Oracle ConText Cartridge*

Oracle Corporation's ConText Cartridge [Oracle, 1999] is a complete information retrieval system built atop Oracle Corporation's Oracle RDBMS. The system includes facilities for: (i) automatic indexing of documents; (ii) document categorization into *themes*; (iii) document summarization, to create document *gists*; (iv) ad-hoc retrieval of documents based on textual content or presence of themes; and (v) viewing of documents with markup of query terms to highlight hits in context.

---

[18] Available: http://doc.altavista.com/business_solutions/search_products/search_intranet/specs_indexing.shtml This documentation describes AltaVista Search Intranet, a commercial application that is functionally identical to AltaVista Search, but designed for intranets (local area networks) rather than the Web. It is suspected (but cannot be verified easily) that AltaVista Search and AltaVista Search Intranet use similar indexing technology.

[19] Available: http://www.lycos.com/

[20] Available: http://www.northernlight.com/

Storage of documents may be internal to the Oracle database tables or external to the tables, as content accessible through the file system or the Web. ConText supports multilingual document bases. Documents may be stored in several formats: as plain text, HTML, Adobe's Portable Document Format, and several others for compatibility with common word processing packages.

The indexing technology uses a conventional inverted index with a user-specifiable stoplist. Granularity is to the character level: the index stores, for each indexing term, the byte-level offsets into the original documents for each occurrence of the indexing term. Information is recorded on stopwords occurring adjacent to indexing terms, to allow phrase retrieval with phrases containing stopwords.

Document categorization uses statistical methods and heuristics, in conjunction with a proprietary 200,000-item knowledge base, called the ConText Knowledge Catalog, defining a lexicon that recognizes over one million English words and phrases. This knowledge base organizes its content into a conceptual taxonomy of 2,000 categories. Categorizing a document results in its assignment under, at most, 16 most-specific categories, called *themes*. Membership in a category implies membership in all higher-level categories in the conceptual taxonomy. Use of these *themes* provides ConText's concept retrieval abilities.

Retrieval is based on Boolean, statistical, and linguistic methods. The first two methods use the inverted index; the last one is based on the generated themes. The query language follows an extended Boolean model and supports the standard Boolean operators, phrase searching, wildcards (which subsume truncation), and DBMS-style fielded queries on structured information. Additional operators request query expansion. This query expansion performs word conflation through stemming, morphological similarity using Soundex or fuzzy matching of the term's spelling, and thesaural expansion using any of the ISO2788 relationships. Relevance ranking may be influenced using score adjustment operators based on the presence, absence, proximity, and manual weight re-assignment of terms.

As a performance enhancement for query refinement, ConText supports stored query expressions, which record the query and its results, for further processing.

ConText supports distributed operation: Several ConText servers and several Oracle database servers may co-operate for true parallel execution.

ConText's services are partitioned into several classes, called *personalities*. Examples include the loader personality for gathering, the indexing personalities (DDL and DML), the query personality for retrieval, and the linguistic personality for document summarization and categorization. ConText is meant to be an extensible framework[21], in the sense that Oracle Corporation may easily improve or add services to the system.

#### *2.3.12.3 Web Information Collector (WIC)*

Prior research by the LAKE group resulted in the development of the Web Information Collector (WIC) [Zayour, 1997]. This tool acts as an intermediary between the user and a Web search engine. WIC sends queries to the search engine and retrieves the Web documents into a local repository. The local repository is then indexed, using Glimpse, and used as a document base for further ad-hoc retrieval. The advantages are that (i) subsequent retrieval and navigation, operating on a smaller, local document base, is faster and more efficient, and (ii) since the local repository already contains (hopefully) relevant documents, precision may be increased. In effect, the document base acts more as a corpus rather than a completely heterogeneous collection of documents.

#### *2.3.12.4 Web Directories*

At this point, one should mention another, very popular way of finding resources on the Web: the Web directory. The canonical example is Yahoo![22] It is a hypertext table of contents with links to Web sites related by subject. Unlike search engines, resource discovery is done manually by librarians that surf the Web in search of new material. This has beneficial impact on precision, but the recall levels are so far from exhaustive that

[21] Oracle ConText is *not* a framework for end-user customization of the retrieval system.

[22] Available: http://www.yahoo.com/

Web directories are useful only for very broad information needs, or otherwise merely to begin navigation.

Given their use of an ontological classification, web directories are thus more related to knowledge management systems, although they do provide a query[23] interface to quickly find subject categories (without the need for lengthy navigation through the taxonomy).

## 2.4 Corpus Linguistics

Recently, in the field of **corpus linguistics (CL)**, particular emphasis has been placed on the exploration of large machine-readable corpora using computer assistance for statistical analysis and result visualization [McEnery & Wilson, 1996]. The topics discussed in this section have been implemented in such software tools as the Text Analyzer (to be discussed later) and WordSmith Tools [WordSmith Tools, 1999].

Please note the specific use of the term *corpus*[24], which implies that the documents of which it is composed are representative of a single domain or language. This is unlike information retrieval, which must often handle a far more heterogeneous document base.

Sometimes, corpus linguistic research begins with the markup[25] of texts [McEnery & Wilson, 1996] to include, beyond formatting attributes and identifying information such as authorship, other data such as linguistic part of speech tags, syntactic structure, and discourse information such as explicit discourse segmentation. This markup facilitates various kinds of analysis, for example, limiting research exclusively to all nouns.

---

[23] For this reason, Web directories are sometimes confused with search engines.

[24] The word *corpus* sometimes carries the connotation of being a standard reference (for example, the Brown Corpus). Within corpus linguistics, corpora are often implicitly assumed to be in a machine-readable form. [McEnery et al., 1997]

[25] Standardization of the markup exists as part of the Text Encoding Initiative (TEI), based on Structured Generalized Markup Language (SGML). TEI is now continued and overseen by the TEI Consortium.
Available: http://www.tei-c.org/

The mention of statistics requires some further explanation. In order to apply statistical techniques with any rigor, certain assumptions must be made: implicitly, a model of language is defined. Common but surprising assumptions that have been employed unjustifiably [Church & Mercer, 1993; Dunning, 1993] in the literature include: the existence of asymptotic normality, or worse, that words occur independently of their context and according to a fixed probability, thus enabling word occurrence to be modeled as a Bernoulli process. Despite the above, useful results have been derived; it's just that statistics are used inappropriately to value the methods above ad-hoc or heuristic techniques.

### 2.4.1 Visualization: Concordances and Word Lists

The primary visualization technique of the corpus linguist is the concordance, and the tool to provide this visualization from source corpora is called a concordancer.

A concordance is a list of words in context. Although this includes words displayed within their sentence, paragraph, or entire document, the most popular concordance is the **Key Word In Context (KWIC)** display (as will be seen in Figure 12 and Figure 34 of the following chapters), which presents all occurrences of a specified keyword, sometimes called the node, one per line and centered, with a fixed amount of contextual text to the left and right. This enables the study of a particular word, observing its use and possible meanings.

Of course, the linguist must decide on which words to study. Here, a word list is of great benefit. The list may be sorted by order of appearance, by frequency of word occurrence, or alphabetically. Although this latter case of an alphabetically sorted list of words may naturally cluster inflected forms that are distinguished solely by suffixes (or prefixes, sorting words as if spelled in reverse), it is not a general solution to the problem of morphological variation. Instead, lists of lemmas or lexemes are often used as an adjunct, or else the list may include all words in their various inflections, but grouped under such headwords.

### 2.4.2 Term Extraction

Corpus linguistic studies rely significantly on frequency statistics. For this discussion, let us distinguish types and tokens. Types may be distinct words, parts of speech (noun, verb), lexemes, or other word classes; tokens are the instances of the types. Note that when we talk about *words*, we sometimes mean distinct words with different spellings, and at other times we mean word occurrences within a document. To avoid ambiguity, we may refer to word-types and word-tokens, respectively. To clarify, the sentence "A can can hold water." has[26] five word-tokens but only four word-types. When one speaks of the frequency of a word, one means the ratio of the number of word-tokens of a given word-type to the total number of all word-tokens (of all word-types).

A term is a word or multi-word group that has an unambiguous meaning in some specific domain. Linguists, particularly terminologists, seek their identification and study, with corpora being a primary source of data.

For single word terms, the word list provides the term candidates, with possible emphasis on high frequency words (or those with high resolving power as determined by tf·idf scores, as in IR). Further pruning of the list may be achieved by noting that the distributional properties of a word that is a term, as used in a specialized domain, will be different than those of the same word in a general language corpus. Thus, a simple technique is to compare word frequencies from both corpora, and use a statistical hypothesis test (such as a t-test). Alternatively, studying the collocational behavior of a word provides further support. Morphological analysis can also be used for term identification, by noting that terms contain typical affixes[27] [Heid et al., 1996].

Multi-word term identification relies on finding collocations, which are discussed below. As it happens, most terms are composed of multiple words [Aussen-Gilles et al., 1995, cited in Kavanagh, 1995].

---

[26] Note that the sentence has five lexeme-types, since there are two senses of *can*.

[27] Morphological analysis is language dependent, of course; we assume the language (such as English) uses affixes.

### 2.4.3 Co-Occurrence Statistics

When computing co-occurrence statistics, the natural question is "What constitutes an instance of a co-occurrence?" It could be two words occurring in the same document, the same paragraph, the same sentence, or within $n$ words, to name a few cases. Computing co-occurrence statistics begins with a choice of node word, around which it is possible to build a concordance of some specified span. The length of span is dictated by the application, but in what follows, the interest lies in tight associations that tend not to cross sentence boundaries. For practical reasons, including implementation efficiency, Barnbrook [1996] suggests using a distance of ±5 words around the node, noting that empirical evidence suggests that most lexical relations can be found within this fixed distance.

Having found co-occurrences, it is necessary to decide which are significant. The co-occurrences are usually scored (Table 5) using mutual information, z-scores of the Normal distribution, t-scores of the Student distribution, or other (sometimes ad-hoc) methods. With many of these techniques, the probability $\mathrm{P}(W_i)$ of a word $W_i$ occurring, independently of the others, is estimated from the observed frequency of that word, e.g. $\hat{\mathrm{P}}(W_i) = N_i / N$ where $N_i$ is the number of occurrences of a word $W_i$ in the corpus (number of word-tokens of that word-type) and $N$ is the total number of word-tokens (of any word-type) in the corpus. Consider a concordance around a node word $W_i$ and let $N_s$ be the number of word-tokens in all spans, or about $10N_i$ using a fixed span of ±5 words. Furthermore, let $N_{jli}$ be the observed number of occurrences of a word $W_j$ within the spans. Using assumptions on word distributions, the expected number of occurrences of $W_j$ within the spans, $\mathrm{E}[N_{jli}]$, will be proportional to the size of the span and the probability of occurrence within the corpus, from which one may derive the estimate $E_{jli} = \hat{\mathrm{E}}[N_{jli}] = N_S \hat{P}(W_j) \approx 10 N_i N_j / N$. Finally, the standard deviation is calculated using $\sigma = \sqrt{N \mathrm{P}(W_j)(1 - \mathrm{P}(W_j))}$ or an approximation thereof [Church & Mercer, 1993].

| Mutual Information Scores | z-scores | t-scores |
|---|---|---|
| $I(W_i;W_j) \equiv \log_2 \frac{P(W_i,W_j)}{P(W_i)P(W_j)}$<br>$I_{i,j} = \log_2 \frac{N_{jli}}{E_{jli}}$ | $z_{i,j} = \frac{N_{jli} - E_{jli}}{\sigma} = \frac{N_{jli} - E_{jli}}{\sqrt{N\left(\frac{N_j}{N}\right)\left(1-\frac{N_j}{N}\right)}}$ | $t_{i,j} = \frac{N_{jli} - E_{jli}}{\sigma} \approx \frac{N_{jli} - E_{jli}}{\sqrt{N_{jli}}}$ |

*Table 5: Scoring of Co-Occurrence Significance*

A cutoff for significance is determined arbitrarily or by picking a confidence level (such as 95%) and using statistical hypothesis testing (z-test, t-test, chi-squared, and others). Note that these tests may have to contend with a very small sample size: measuring co-occurrences, such as pairs of words, implies a magnification of the consequences of the asymmetric frequency characteristic that a few common words form the bulk of a text, and the large number of low-frequency words form the remainder [Rundell, 1996].

In fact, it is hard to justify the normality assumption that underlies the above statistics. Thus, other statistical tests with weaker assumptions than those above have been used. Dunning [1993] presents a bigram ranking technique that uses a Log-Likelihood Test. The author further indicates the need to explore other, distribution-free statistical tests. For example, he cites Fischer's exact method.

### 2.4.4 Collocational Analysis

Benson [1990, cited in Smadja, 1993] defines a collocation as "an arbitrary and recurrent word combination". It has four important properties [Smadja, 1993]: it is *arbitrary* (existing in some languages and dialects but not others), *domain-dependent* (such as technical jargon), *recurrent* (in that it is used repeatedly in a given context), and forms a *cohesive lexical cluster* (the presence of part of the collocation often implies the rest).

A collocations is thus more restricted than a mere word co-occurrence.

Smadja further classifies collocations into three kinds: predicative relations (words used in similar syntactic relations), rigid noun phrases (fixed sequences of words),

and phrasal templates (idiomatic phrases with slots that must be filled in). Smadja's Xtract [Smadja, 1993] is a software tool that identifies collocations through statistical techniques, similar to the co-occurrence statistics that were presented previously. In a first stage, Xtract identifies word bigrams that are collocations; thereupon, a second stage uses concordances containing these word bigrams to find collocations of more than two words.

Useful for term extraction, collocations can also be used to find, for example, which nouns and verbs go together, such as the verb-noun collocations *fly-airplane* and *drive-car.*

When the interest is not on the associated words themselves, but on the syntactic surroundings of word and language structure in general, then a similar analysis can be performed using word classes (syntactic tags, part-of-speech tags, etc.). The result of this analysis is called a colligation rather than a collocation.

## 2.5 Knowledge Management and Engineering

The difference between noise, data, information, and knowledge[28] [Giarratano & Riley1994; Rutkowski & Stasko, 1999] is essentially one of degree of meaning and utility. Noise is without interest, and obscures data. Data is potential information in a raw form, as random facts, some of which may be useful and some of which may be irrelevant or redundant. The application of data relevant to some purpose transforms it into information. When information is applied with human experience or in the context of some intelligence, it gains meaning and becomes knowledge.

In the context of a knowledge management system, such as an expert system, one may distinguish further between *stated knowledge,* which is the set of facts and rules entered into the system, and *derived knowledge*, which is the set of facts and rules that are derived (such as via logical deduction) from others, rather than explicitly stated. The derivation of this knowledge is called *inferencing*.

A **knowledge base (kb)** is a repository for stated knowledge.

[28] The meaning of the word *knowledge*, as it is used in the information technology industry, is rather varied. To quote, "No two people are talking about knowledge in the same way" [Lucier, C., cited in Rutkowski & Stasko, 1999].

### 2.5.1 Knowledge Representation

The choice of knowledge representation formalism impacts what can be represented and how efficiently it can be processed.

#### *2.5.1.1 Object-Attribute-Value Triplets*

With an Object-Attribute-Value (OAV) representation, each triplet is a fact about some specific object or subject. Each fact describes some property about the object, for which there is an associated value. Examples of OAV triplets are: `(car, is-a, vehicle)`, `(car, has, wheels)`, `(myCar, instance-of, car)`, `(myCar, has, wheels)`, `(myCar, color, blue)`, `(myCar, owner, John)`, and `(John, instance-of, person)`.

#### *2.5.1.2 Semantic Networks*

An associative network [Crestani, 1995] is a graph, with nodes representing information items and links defining relationships between the nodes. The nodes are associated via links that may be directed or undirected and labeled or unlabeled. A semantic network[29] [Quillian, 1968, cited in Crestani, 1995; Sowa, 1995; Giarratano & Riley, 1994] is an associative network with directed and labeled links, in which the nodes represent concepts that may be related in a taxonomy through *is-a* or *instance-of* links. The meaning of the latter two links are that one concept (the *species*) is a special case of the superconcept (the *genus*), inheriting all of its attributes, but with possible alterations or additions (the *differentia*). A semantic network can be viewed as a set of OAV triplets. Thus, the example presented in the section above can be rewritten as a semantic network (see Figure 5); note the use of inheritance renders the triplet `(myCar, has, wheels)` redundant.

[29] The term *semantic network* is sometimes used inappropriately in place of the terms *associative network* or *graph*, particularly in the IR literature.

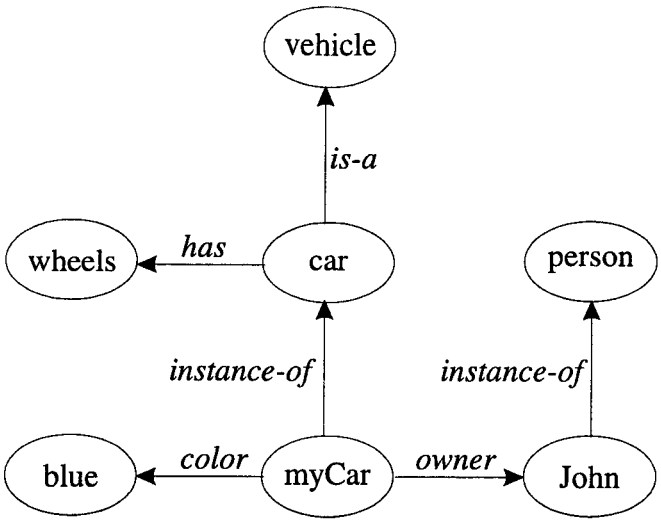



*Figure 5: Example of a Semantic Network*

### *2.5.1.3 Frames*

A frame is "a data structure for representing a stereotyped situation [... and can be thought of] as a network of nodes and relations" [Minsky, 1975, cited in Sowa, 1994]. A frame describes a type or an instance, and is composed of a set of *slots* that are each to be filled in with some value; this value may be subject to possible constraints specified by the frame. Thus, each slot is an attribute-value pair. In many cases, the value of a slot, called the *slot-filler*, is a *facet*, which is itself a value-attribute pair. As a further extension, slots may contain default values, pointers to other frames, and procedural knowledge that may be executed. In this way, a set of frames can be viewed as a semantic network with extensions. Figure 6 presents the previous example (with some changes) as frames. In this example, THING is used to represent a single, top-level concept.

| **vehicle** | |
|---|---|
| is-a | THING |

| **car** | |
|---|---|
| is-a | vehicle |
| hasPart | type: wheel<br>quantity: integer |
| color | type: Color |

| **myCar** | |
|---|---|
| instance-of | car |
| color | blue |
| owner | John |

| **person** | |
|---|---|
| is-a | THING |

| **John** | |
|---|---|
| instance-of | person |

*Figure 6: Example of Frames*

#### *2.5.1.4 Logic Statements*

Logic is a formal language for expressing knowledge precisely, allowing inferencing by logical deduction. Many kinds of logic exist; some examples include:

- *Propositional logic*: A proposition is a statement that must be true or false (it cannot be both or neither). In propositional logic, several operators such as Boolean conjunction, disjunction, and negation may be applied to link propositions together and make more complex propositions. Propositional logic does not include the notion of variables.
- *Predicate calculus or first-order logic*: Similar to propositional logic, predicate calculus includes variables (specifically, terms) and quantification operators.
- *Typed logic*: This is a logic where variables are given a type, which specifies that allowable values must be from some specified set.

#### *2.5.1.5 Production Rules*

In a production-rule based system, knowledge is encoded as *if-then* rules [Giarrratano & Riley, 1993; Luger & Stubblefield, 1992]. Each *if-then* rule includes a premise which, when it is true, will imply the truth of the conclusion. The premise and the conclusion can take the form of logic statements or, more simply, can be OAV triplets whose existence within the knowledge base is being queried or asserted.

### 2.5.2 Knowledge Representation Languages

Many languages exist for representing knowledge. For example, logic statements may be represented using conventional mathematical notation, as Horn clauses, or as PROLOG statements, among others. Knowledge Interchange Format (KIF) [NCITS.T2 Committee On Information Interchange And Interpretation, 1999a] and Conceptual Graph (CG) notation [Sowa, 1995; Delugach, 1999; NCITS.T2 Committee On Information Interchange And Interpretation, 1999b] are proposed ANSI standards for information interchange. Ontology Markup Language/Conceptual Knowledge Markup Language (OML/CKML) are XML based knowledge representation languages [Kent & Neuss, 1997]. Simple HTML Ontology Extensions (SHOE) [Parallel Understanding Systems Group, 1999a] is another knowledge representation language, influenced by the Web and

designed to embed formal knowledge into existing HTML documents, using additional markup.

### 2.5.3 Frequently Asked Questions (FAQs)

FAQs are compendiums of frequently asked questions, along with their answers, expressed in natural language. While they are not knowledge bases, lacking the required formality, they do serve a useful purpose as repositories for (ideally) important and concise information, as collected by a domain expert. FAQs are often available as plain text or hypertext documents, requiring the user to browse for relevant information. Systems such as the EKD QA System [Sneiders, 1998], Auto-FAQ [Whitehead, 1995], FAQ Finder [Burke et al., 1997], and ExtrAns [Mollá, 1999] allow more efficient retrieval of relevant answers, often using a natural language question answering interface to a conventional information retrieval system. As, in most FAQs, the document base is often much smaller than in conventional IR applications, and the questions tend to follow typical linguistic templates, such as "What is the definition of X", natural language processing techniques become more feasible.

### 2.5.4 Databases and Frame-Based Knowledge Representation Systems

Karp and Paley [1995] criticized knowledge representation systems for failing to address two important issues with respect to their scalability: inefficient support of large knowledge bases and lack of concurrent multi-user access. They asserted that, at the time of their research, all existing frame representation systems required their knowledge bases to be loaded entirely into main memory. Instead, they advocated the use of database management technology, and proceeded to implement their ideas in two systems, LOOM and THEO. They then analyzed the performance impact of frame faults, that is, access to frames that are not in memory and hence must be loaded from secondary storage. Given the assumption that only a small subset of the knowledge base may be viewed or updated at any given time, they found their experiments successful.

PARKA [Parallel Understanding Systems Group, 1999b] is another project addressing the scalability of knowledge representation systems, aiming to allow efficient use of knowledge bases exceeding millions of assertions in size. Earlier work emphasized

massive parallelism to achieve this goal; more recent work, as part of PARKA-DB, focuses on the utilization of database management technology.

### 2.5.5 Knowledge Management Systems

This thesis will limit the presentation of the many knowledge management systems to a small sampling. In addition, let us very briefly mention the High Performance Knowledge Bases (HPKB) research program. HPKB aimed to "advance the technology of how computers acquire, represent and manipulate knowledge" [HPKB, 1999] with emphasis on "the technology needed to enable system developers to rapidly (within months) construct large (100K-1M axiom/rule/frame) knowledge-bases that provide comprehensive coverage of topics of interest, are reusable by multiple applications with diverse problem-solving strategies, and are maintainable in rapidly changing environments" [HPKB, 1999]. HPKB presented several real-world challenge problems for information systems, and leveraged existing tools, such as LOOM and Cyc (described below), in solving these. The HPKB project was recently completed, and a new project started, called Rapid Knowledge Formation (RKF). RKF continues HPKB, with the goal of enabling "distributed teams of subject matter experts (SMEs) to enter and modify knowledge directly and easily, without the need for specialized training in knowledge representation, acquisition, or manipulation." [RKF, 2000] As such, RKF overlaps with the research of this thesis.

#### *2.5.5.1 CODE4 and its Successors*

Conceptually Oriented Design/Description Environment version 4 (CODE4) [Skuce, 1995a; Skuce & Lethbridge, 1997] is an interactive, multi-functional, generic frame-based knowledge management tool aiming to "[amplify] human intelligence rather than [...] to run autonomously, [... and to] assist average computer users to store and communicate knowledge more effectively" [Skuce & Lethbridge, 1997]. Emphasis is placed on providing a rich user interface, favoring expressiveness over rigid formality within the knowledge base. CODE4 and its predecessors have been used effectively in several applications, from education to software engineering [Skuce, 1995a; Skuce & Lethbridge, 1997].

In CODE4, the basic units of knowledge are about a *concept*, which is the basic knowledge structure that participates in the underlying inheritance mechanism. The authors of CODE4 distinguish between a concept, as a piece knowledge in a CODE4 knowledge base, and the thing that the concept represents, in the real (or imagined) world [Skuce, 1995a]. To record meta-level information about a concept itself, rather than the thing represented, CODE4 features *metaconcepts*, which are created automatically as needed.

The name of a concept, called a *term*, is itself a concept that exists as part of a lexicon, allowing the knowledge base to record linguistic information such as synonyms or part of speech information.

CODE4 statements, the basic unit of knowledge, correspond to frame/slot/slot-filler triplets consisting of a *subject*, a *predicate*, and, optionally, a *value*. Statements and predicates are treated as concepts, so they also participate in the inheritance hierarchy, can have statements about them, and can be referenced in other statements. A statement about a statement is called a *facet*.

In total, CODE4 has five kinds of hierarchies:

- The inheritance hierarchy defines a hyponymic taxonomy. This hierarchy has a single most general concept, called *thing* by default.
- The property hierarchy is often used for grouping properties into categories, but may also be used to define entailment relationships. For example, walking entails moving.
- Each concept has a statement hierarchy consisting of the statements that apply to the concept, as a result of inheriting a sub-hierarchy of the property hierarchy.
- Relation hierarchies are defined by following chains of concepts via the values of statements of one or more properties. That is, a relation hierarchy can be defined by the associative network represented by a subset of statements.

- Facet hierarchies are defined by the recursive nature of statements made about statements.

CODE4 encourages, but does not require, adherence to certain syntactic conventions, including the use of a constrained English language called ClearTalk [Skuce, 1992; Skuce & Lethbridge, 1997; Skuce, 1999b] when entering statement values. A subset of ClearTalk is machine-interpretable, in the sense that expressions using this subset carry formal semantics, which may be used during inferencing or constraint checking. Values that are not interpretable are merely stored for later display.

The CODE4 user interface allows the user to request several kinds of displays, with various filters to limit the information that is presented.

- An outline browser displays hierarchical relations using different levels of indentation.
- A graphical browser displays information as a semantic network.
- A property matrix browser displays a grid with concepts along one axis, and properties along the other. Each cell displays the value (complement) corresponding to the intersection of the property with the concept. Such a browser is useful for comparing concepts and identifying differences between them.

CODE4 is implemented in Smalltalk. The knowledge base is maintained entirely in memory as Smalltalk objects. The user interface manipulates the knowledge base using an abstraction called a *knowledge map*, which defines an associative network of concepts starting from an initial concept and recursively following specified relations. For use by external applications, CODE4 defines a socket-based protocol allowing its use as a knowledge server across the Internet.

In 1995, the LAKE group, including the author of this thesis, undertook the development of a successor to CODE4, called the Knowledge Processor, version 1 (KP1). In order to allow concurrent multi-user access to large, shared knowledge bases, KP1 used relational database management technology, specifically, the Borland Database

Engine, as part of the Paradox RDBMS. Ideas from KP1 served as the foundation for the group's future work on knowledge management systems.

However, with the increasing popularity of the Internet and, at that time, the recently born Web, the KP1 project was abandoned, favoring a new direction towards Web accessibility that resulted in the Intelligent Knowledge Acquisition, Retrieval, and Understanding System (IKARUS) [Skuce, 1997]. IKARUS featured an HTML-based design using a Common Gateway Interface (CGI) server to generate Web pages on demand. As a result, the knowledge base could be accessed and manipulated worldwide from any appropriate Web browser.

The interface included a menu of features at the very top, an outline view of the conceptual taxonomy on the left, and a formatted presentation of statements on the right. All occurrences of subjects within the lexicon were rendered as hypertext links. Also, as a consequence of the HTML design, complements could contain arbitrary HTML markup, including links to embedded images or audio, or to navigate to any other Web page, including subjects within other IKARUS knowledge bases.

The CGI server was implemented as Perl scripts on a Unix machine, and employed Berkley's dbm, a hashing-based persistent storage library, to maintain the knowledge base in secondary storage. As with database technology, this allowed large knowledge bases by avoiding the need to always load the entire knowledge base into main memory. Reliance on HTML, CGI, and JavaScript limited[30] the flexibility of the user interface and imposed some efficiency concerns due to the latency of Internet communications. However, with the advent of Java[31], a complete Web-based applications development environment, it became possible to overcome the problems. For example, a

[30] HTML, as opposed to Dynamic HTML (DHTML), is static: a Web page cannot be dynamically modified, instead requiring the Web page to be regenerated each time there is an update. CGI, without *cookies*, is stateless, that is, cannot record state information across multiple transactions. JavaScript has limited user interface capabilities.

[31] Java, including the Java Language, the Java Platform, the Java 2 Platform, the Java Software Development Kit (SDK) for the Java Platform (JDK1.1.x) and Java 2 Platform (JDK1.2/1.3), Core Java Class Library and Standard Extensions, including Java Database Connectivity (JDBC), are all the product of Sun Microsystems. Available: http://www.javasoft.com/

graphical view of the conceptual taxonomy was added as a Java applet[32] that was implemented by the author of this thesis.

#### *2.5.5.2 Cyc*

The Cyc Project [Lenat & Guha, 1989; Lenat et al., 1990; Lenat, 1995; Cycorp, 1999] aims to solve problems of the *software brittleness bottleneck* and the *knowledge acquisition bottleneck* encountered in Artificial Intelligence (AI).

Brittleness is the inability to operate in unexpected situations: brittle software, when faced with unexpected environments, will fail completely rather than degrade gracefully. This bottleneck, which lies in the difficulty of creating software that is not brittle, limits the competence and usability of current software.

The knowledge acquisition bottleneck results from the difficulty of acquiring and presenting information in a form suitable for machine-processing. At present, knowledge acquisition is labor intensive, often requiring the services of an expert knowledge engineer.

Predicated on the belief that brittleness results from a lack of common-sense knowledge, Cyc's solution begins with the assembly of an "immense multi-contextual knowledge base" [Cycorp, 1999], "spanning human consensus knowledge" [Lenat et al., 1990]. This knowledge base features a very large ontology of the world, and contains "over 1,000,000 hand-entered assertions" [Cycorp, 1999]. Cyc is further comprised of "an efficient inference engine, a set of interface tools, and a number of special-purpose application modules" [Cycorp, 1999]. It is further assumed that the reuse of existing knowledge will ease future knowledge acquisition tasks. It is hoped that bootstrapping with a sufficiently large knowledge base may ultimately allow automated knowledge acquisition from such sources as natural language texts.

The Cyc knowledge base is expressed using the CycL representation language. The CycL language was originally described [Lenat & Guha, 1989] as being, in part,

---

[32] The Graph Viewer applet was later integrated into DocKMan's Knowledge Base Builder and the Knowledge Organizer.

frame-based, with additional support of predicate calculus for situations where frames lack sufficient expressive power. It is noted that a great amount of knowledge can be expressed as frames, and that such a representational model offers greater simplicity and computational efficiency. Within CycL, elements such as slots can themselves be represented as frames. Furthermore, each slot has several *fields* to record audit tracking information and belief values, including who believes an assertion and with what truth value. Assertions sharing common assumptions can be grouped together, forming *microtheories*. Cyc supports uncertainty and contradiction using these belief values and microtheories. Truth values use a five-point scale for certainty factors: monotonically false, default false, unknown, default true, and monotonically true. Monotonical truth values are stronger than default truth values. The former result in contradictions that must be resolved, whereas the latter merely imply that an assertion is true or false unless there is another assertion that contradicts it. As well, an assertion may be explicitly declared as stronger or weaker than another.

More recently, the Cyc authors emphasize that CycL is not frame-based. It is "essentially an augmentation of first-order predicate calculus with extensions [...] and second order features", and the Cyc knowledge base should be viewed as a "sea of assertions" [Cycorp, 1999]. Cyc *terms*, corresponding to frames in a frame representation, are related through Cyc *relations*, which include *predicates* and *functions*. Predicates have truth values, whereas functions return arbitrary values.

The knowledge base includes information about various concepts, as well as a lexicon of information about the words used to represent concepts, to aid in applications such as natural language understanding.

The inference engine performs logical deduction, with many heuristics and special cases, such as to handle inheritance, for increased efficiency.

The primary user interface to the Cyc knowledge base is based upon dynamically generated HTML. Each HTML page displays a Cyc term and all of the assertions in which it is involved. Every occurrence of a Cyc term is itself a hypertext link to its HTML page, for efficient navigation. The interface allows searching and editing the

knowledge base, and posing queries to the inference engine. Other interface tools include a hierarchy browser, which displays any desired subtree of the conceptual taxonomy in outline format.

#### *2.5.5.3 Web Analysis and Visualization Environment (WAVE)*

With respect to the exploration of knowledge bases, the Web Analysis and Visualization Environment (WAVE) Conceptual Browser [Kent & Neuss, 1997] is a tool allowing navigating between concepts, using several panes that show related information based on intensional and extensional similarity to the chosen model concept. Intensional similarity implies concepts or instances sharing the same or similar properties, regardless of the instances they have in common. Extensional similarity implies concepts having the same or a similar set of instances.

# Chapter 3: TA, KO, and Related Systems

Having presented the conceptual background, it is now possible to provide very concise summaries of several independent systems that were integrated as subsystems into TAKODO. TAKODO and all of the following systems are part of the broader LAKE project, called DocKMan [Skuce, 1998; Skuce, 1999a], aiming to integrate document management with knowledge management.

## 3.1 Historical Background

At the inception of TAKODO, the goal was to integrate a text analysis system with a knowledge management system. Originally, DocKMan's Knowledge Base Builder (DKBB), an existing system to be described later, was to be TAKODO's only knowledge management system. However, difficulties were encountered during the integration process, due to several limitations of DKBB and its poorly structured architecture, which complicated the adaptive maintenance process. An attempt was made by the author of this thesis to retrofit[33] DKBB with a new, object-oriented API, while simultaneously addressing some of the more major limitations. As progress continued, it became apparent that this attempt, while adequate for integrating DKBB as an information source, would not satisfy all of TAKODO's needs in knowledge management.

Hence, it was concluded that TAKODO needed another persistent storage mechanism for its knowledge base. The author proceeded to recreate a functionally similar knowledge management core, using the new API. As development continued, the project's complexity was deemed excessive for the specific needs of TAKODO. Existing code was shelved, and the author undertook a new project, TAKODO Candidates, that rejected the goal of functional similarity to DKBB.

[33] In the terminology of design patterns, this is called a *façade*.

Within the LAKE group, there were other attempts at adaptive maintenance applied to DKBB, for different research purposes. Subsequent to the failure of these attempts, the author's shelved knowledge management core was exhumed by members of the LAKE group, who undertook concurrent development of a new user interface atop this core. The result became the Knowledge Organizer (KO). KO quickly replaced DKBB. Thereupon, DKBB became a legacy system. Further development of DKBB has been discontinued, although TAKODO still maintains support for existing DKBB knowledge bases.

Although KO became available very late in the development stage of TAKODO, it was successfully integrated into TAKODO as its third knowledge management subsystem, the others being DKBB and TAKODO Candidates. Due to their common historical origins, TAKODO and KO are tightly integrated and share a common code base. Hence, several features from one system have migrated into the other.

### 3.1.1 DocKMan's Knowledge Base Builder (DKBB)

DocKMan's Knowledge Base Builder [Prpic, 1997a; Prpic, 1997b] is a generic frame-based knowledge management tool with inheritance-based inferencing that was implemented by Jamie Prpic of the LAKE group. DKBB was distinguished from many other knowledge management tools by the three following features:

- The system followed a client-server architecture. The client was written for the Java Platform as a Java applet, thus making it available across the Web, via any Java-enabled browser. Continuing the goals of IKARUS, this allowed collaborative, distributed knowledge management, worldwide.
- The knowledge base was stored and accessed using conventional database management technology, using the Sybase[34] RDBMS. The goal was to allow very large knowledge bases, exceeding available physical memory. A secondary benefit was greater interoperability with other software, since the computer industry already provides much support for database interaction.

[34] Available: http://www.sybase.com/

- There was support for informality and contradiction within the entered knowledge. Users are encouraged to follow certain conventions and use ClearTalk. However, as with CODE4, this was not required, and statements could be entered with any desired character strings. Furthermore, several statements could be made about some property of a given concept, even if the asserted values for the properties were contradictory.

Similar to CODE4, knowledge in DKBB is expressed via *statements*. Each DKBB statement must have a *subject* and a *predicate*. An optional *complement* stores any phrase directly associated with the predicate, as it relates to the subject. Normally, these three parts correspond to the subject, verb, and direct object of a simple sentence.

Additional information may be expressed by associating *facets* with the statement. Each facet is a *facet-name* and *facet-value* pair. These facets are used to record several categories of information: linguistic facets store additional syntactic elements, such as indirect objects or relative clauses; annotation facets store audit information such as source citations; and meta-knowledge facets store information about the statement itself, such as the knowledge source, author, or date.

Subjects, predicates, and facet-names are all *lexemes*, which are textual names having a sense number (the default sense number is 1). In order to match, the name and sense number must be identical, including case sensitivity. On the other hand, complements and facet-values are arbitrary chunks of text.

All subjects must be located within a hyponymic taxonomy used to model the concept hierarchy. It is the sole hierarchical relationship explicitly supported by the system. This taxonomy has a lattice-like structure, with a single top element, called *kbTop*, and with support for multiple inheritance, that is, multiple hypernyms for any given subject. Unlike for statements, the taxonomy does not allow uncertainty or contradiction: (i) every subject, except *kbTop*, must have at least one hypernym; (ii) relationships within the taxonomy are either present or absent, without any notion of certainty factors; and (iii) cycles in the taxonomy are strictly disallowed, so that a subject may not be its own hypernym.

The hyponymic taxonomy controls statement inheritance, which is the mechanism used for inferencing. In addition, each predicate specifies one of three predicate inheritance modes, for use with all statements having that predicate. Similarly, each facet-name specifies one of three facet inheritance modes. These modes influence the inheritance algorithm as follows:

- Full: The statement (or facet) is inherited to all hyponyms, with the predicate (or facet-name) and complement (or facet-value) preserved unchanged.
- Predicate-Only (or Facet-Only): The statement (or facet) is inherited to all hyponyms, with the predicate (or facet-name) preserved unchanged, but the complement (or facet-value) left blank. This inheritance mode is used to specify templates for new statements.
- None: The statement (or facet) is not inherited to any hyponyms.

Figure 7 shows the DKBB user interface. The top-left area shows an outline view representing the hyponymic taxonomy, where branches may be selectively expanded or contracted. The top right area includes the editing area, where new statements are entered and existing statements are modified. The bottom half shows grids for the statements and the facets.

Within this outline view, the user may select a single subject of interest, found either via navigation or via a string searching function. Thereupon, the corresponding statements are shown in the statements grid. These include either (i) all statements about the selected subject, with inheritance from hypernyms, or (ii) all statements about the selected subject or its hyponyms, without inheritance. Optionally, a hiding feature removes from display any inherited statements that are overridden by other statements with the same predicate.

In the statement grid, a single statement of interest may be selected. Thereupon, the corresponding facets are shown in the facet grid, and the statement is made available for editing in the editing area. As well, all statements, regardless of their associated

subject, may be displayed in a separate window offering limited filtering abilities by substring matching on the subject, predicate, and complement columns.

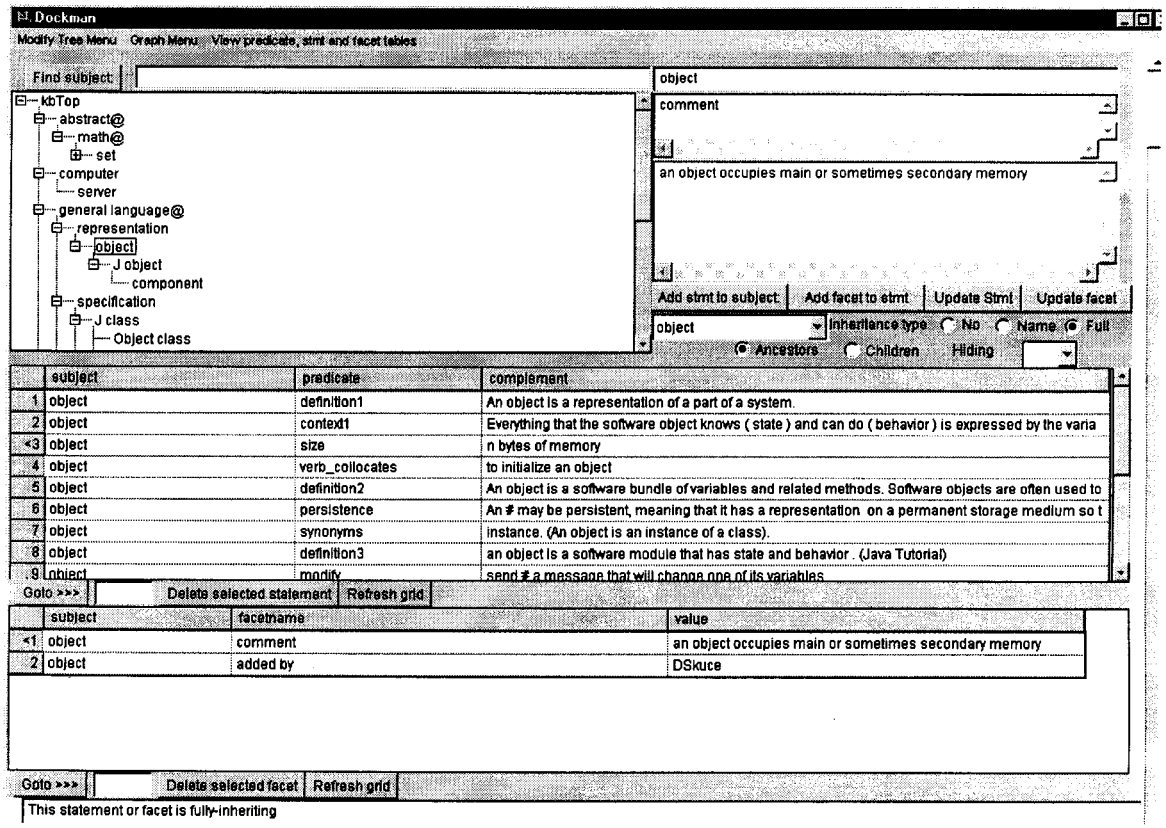



*Figure 7: DocKMan's Knowledge Base Builder*

For ease of navigation and manipulation of the concept hierarchy, DKBB includes a graphical view[35], which models the knowledge base as a semantic network. Figure 8 shows a subgraph of the hyponymic taxonomy. The graphical view has several visualization options, including color indicating depth, node filters to control the extent of subgraphs, and whether to include statements in the semantic network. Concept manipulation functions allow the user to edit the taxonomy, for example, reparenting subjects as needed. The graphical view is dynamically driven by the knowledge base; any changes in the graph are immediately reflected in the rest of the DKBB user interface, and vice-versa.

[35] The Graph Viewer was originally implemented by the author of this thesis, as part of the Ikarus knowledge management system [Skuce, 1997]. It was subsequently ported to DocKMan's Knowledge Base Builder and to the Knowledge Organizer.

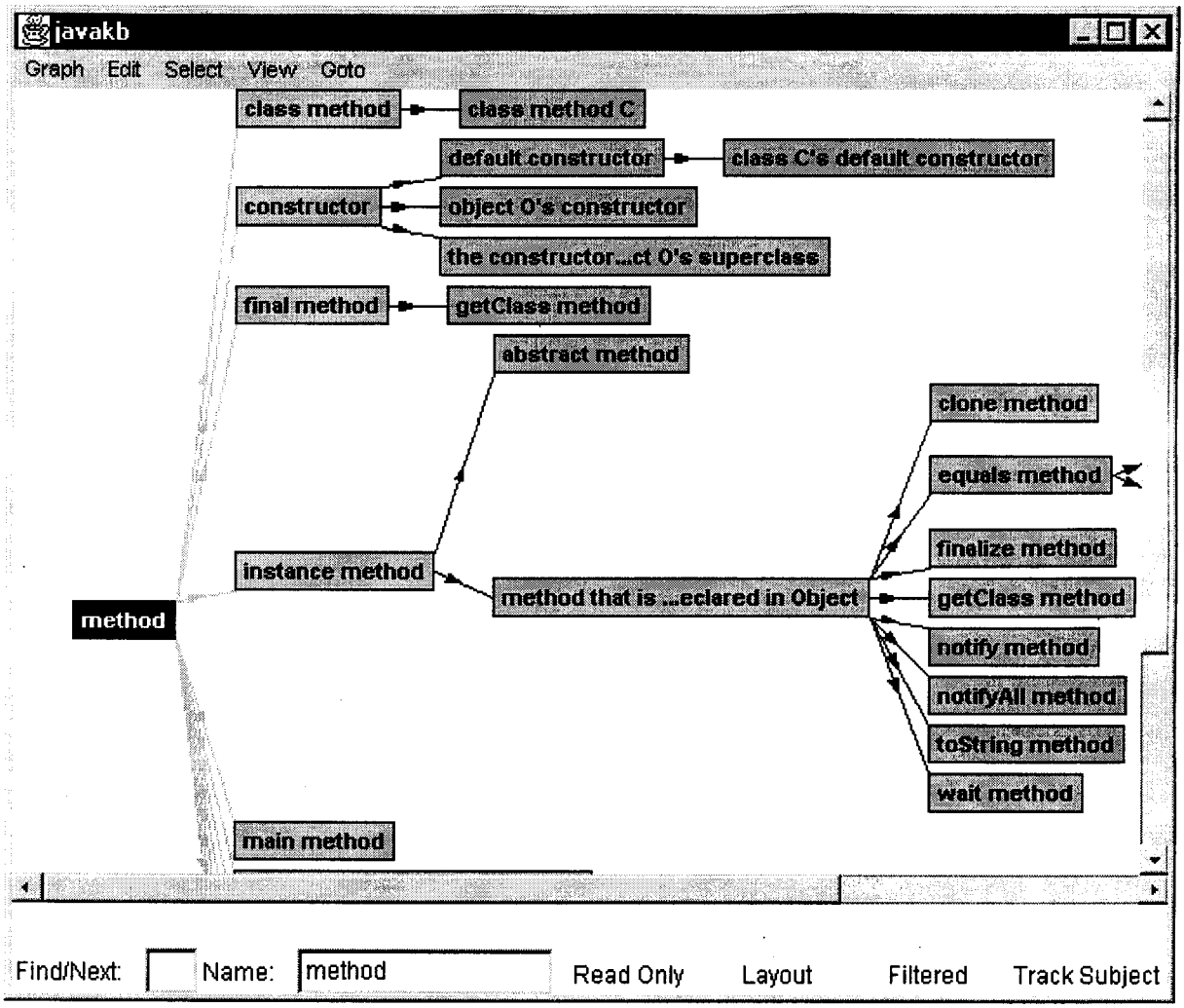



*Figure 8: Graph View of Concept Hierarchy (Ikarus, DKBB, and KO)*

### 3.1.2 Limitations and Design Flaws

DKBB was written as a prototype using new and untested Java technologies. As such, the design and implementation exhibited several major deficiencies:

- Implementation of the system was done using Symantec™ Visual Café[36], a development environment for the Java Platform. DKBB relied on the vendor's DBMS middleware, DBAnywhere, which was found to be problematic, exhibiting a high rate of failure. Furthermore, communication with DBAnywhere used a proprietary database API designed to overcome limitations of the just-emerging

[36] Available: http://www.symantec.com/cafe/

standard database API, called Java DataBase Connectivity (JDBC). The latter evolved, displacing the proprietary interface, which was subsequently discontinued by the vendor.

- The architecture of DKBB was that of a thick-client. That is, most processing occurred on the client side. In particular, the server consisted purely of the DBMS, with little support for knowledge management operations. Constraint checking and enforcement of global consistency was distributed across several clients, rather than centrally managed. As a result, there was significant data transfer between the client and server. A specialized server would have alleviated this problem, transmitting only the results of processing rather than all data needed for such functions as inferencing.

- Performance was also problematic. The thick-client architecture that required numerous information requests to satisfy atomic operations, coupled with the latency of Internet communications, resulted in significant delays[37].

- The current implementation lacked scalability, despite the underlying use of a DBMS. The entire concept hierarchy and all subjects, predicates, and facet-names had to be maintained in memory. Only the statements and their facets were loaded when needed.

- Maintainability and extensibility had been compromised in favor of rapid development and deployment. Figure 42 of the Appendix shows an abridged class diagram of the system. Note the lack of adherence to conventional object-oriented design principles in the modeling of domain data. Although the system was modular, with classes grouped into Java *packages*, there was no use of subclassing, nor evidence of appropriate encapsulation and information hiding. This is witnessed by the lack of visibility modifiers to separate private implementation details from the public interface. Failed attempts at extending the

[37] Operating on a Local Area Network (LAN), most interaction with the user interface complete in less than two seconds. However, delays exceeding 45 seconds have been observed when executed remotely via high-speed (56 kbps) modem through the Internet.

system, such as to add system support for a meronymic hierarchy, confirmed the maintainability issues.

- The system could not be used as a knowledge server. There was no clean API to the knowledge management functions, for use by external applications, although there was an API for access to the user interface (through the classes of the `tables` package). Hence, an external application could select a subject in the outline view, resulting in a display of the associated statements, but there was no way to retrieve these statements for further processing, other than via the database. The external application would have to re-implement its own knowledge management functions, based solely on the database schema.

As a result of the above issues, further development of DKBB was discontinued.

## 3.2 The Text Analyzer (TA)

The Text Analyzer, originally the Master's thesis of Judy Kavanagh as member of the LAKE group, is a tool designed to assist in extracting knowledge from text, with particular emphasis on question answering. The first version [Kavanagh, 1995] was implemented in a Unix environment, using Icon, Smalltalk, and C. A second version [Kavanagh, 1998; Prpic, 1997a] was a complete re-implementation for the Microsoft Windows environment, using Visual Basic. Although functionally similar[38], the user interface was significantly updated. The remainder of this thesis concentrates on the second version, which is the one integrated into TAKODO.

[38] The second version omits some collocational analysis features.

The list of operations performed by the TA is presented in Table 6. Operations are chosen from a main menu. Each operation uses a distinct window, with its own user interface, although the TA maintains consistency across these interfaces. Figure 9 shows the noun frequency list, which presents, in lemmatized form, all words tagged as a noun, excluding stop-words, sorted in decreasing order of frequency. Such a list is useful for term extraction. Figure 10 shows subject-verb-object co-occurrences within the tagged sentences, which may be used to find candidate statements for entry into a knowledge base [Prpic, 1997a].

| Operations | | Description |
|---|---|---|
| Frequency statistics | • Nouns<br>• Verbs<br>• Compound nouns<br>• Column values | Generates word lists from the document base, sorted lexicographically or by frequency. Finds all distinct values from any single, arbitrary column displayed as part of output from other operations. Allows simple filtering by substring matching. |
| Concordances | • KWIC<br>• Verbs following word<br>• Adjective preceding noun | Concordance display of all sentences containing a specified search pattern (a word or a phrase). The search pattern is centered; optionally, the following verb or preceding noun may be displayed in a separate column. |
| Collocations | • Noun-Noun<br>• Subject-Verb-Object | Shows all contiguous NN or SVO occurrences, one per row. Has several fixed and predefined columns corresponding to syntactic structure (premodifiers, postmodifiers, etc.). Allows sorting and simple filtering by substring matching on any column. |
| Question answering | | Answers several types of questions, using proprietary text patterns to find passages with the desired semantic meanings. Output is a displayed as a concordance. |
| Superconcept extraction from definitions | | Information extraction of hypernyms. Limited to a domain-specific corpus that uses *Nortel Standard English*, a constrained English language. |
| *Other document management operations* | | Show greater context of a sentence (e.g. entire paragraph), list of documents, document structure (section headings), etc. |

*Table 6: Functional Overview of Text Analyzer Operations*

The techniques of question answering in TA are based on linguistic and statistical analysis of corpora. Research by the LAKE group categorized common questions, as used in FAQs and other sources, deriving a set of question types (Table 7). Corresponding to these question types, a collection of relevant linguistic patterns, totaling about 50, was

identified. These act as indicators or signal phrases, pinpointing the locations of answers within sentences. For example, the pattern ", `that is,` " is used to identify sentences that may contain the definition of a given search word. Experimental evaluation found the system to have an average recall of 83% and a precision of 84% [LAKE Research Group, 1999].

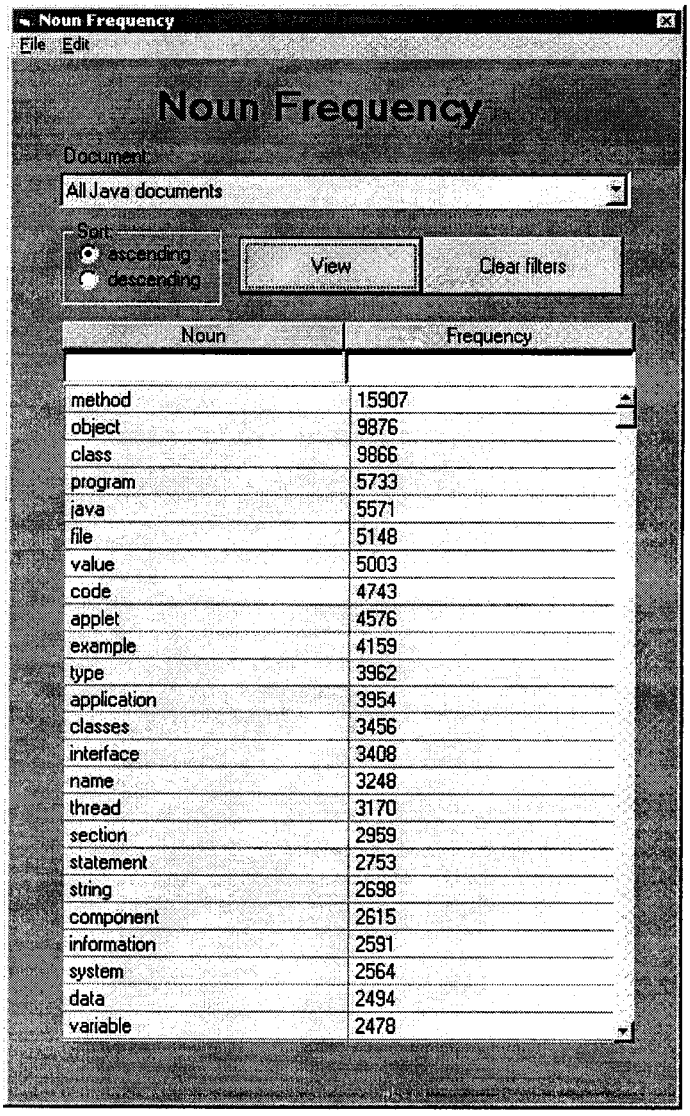



*Figure 9: Noun Frequency using TA*

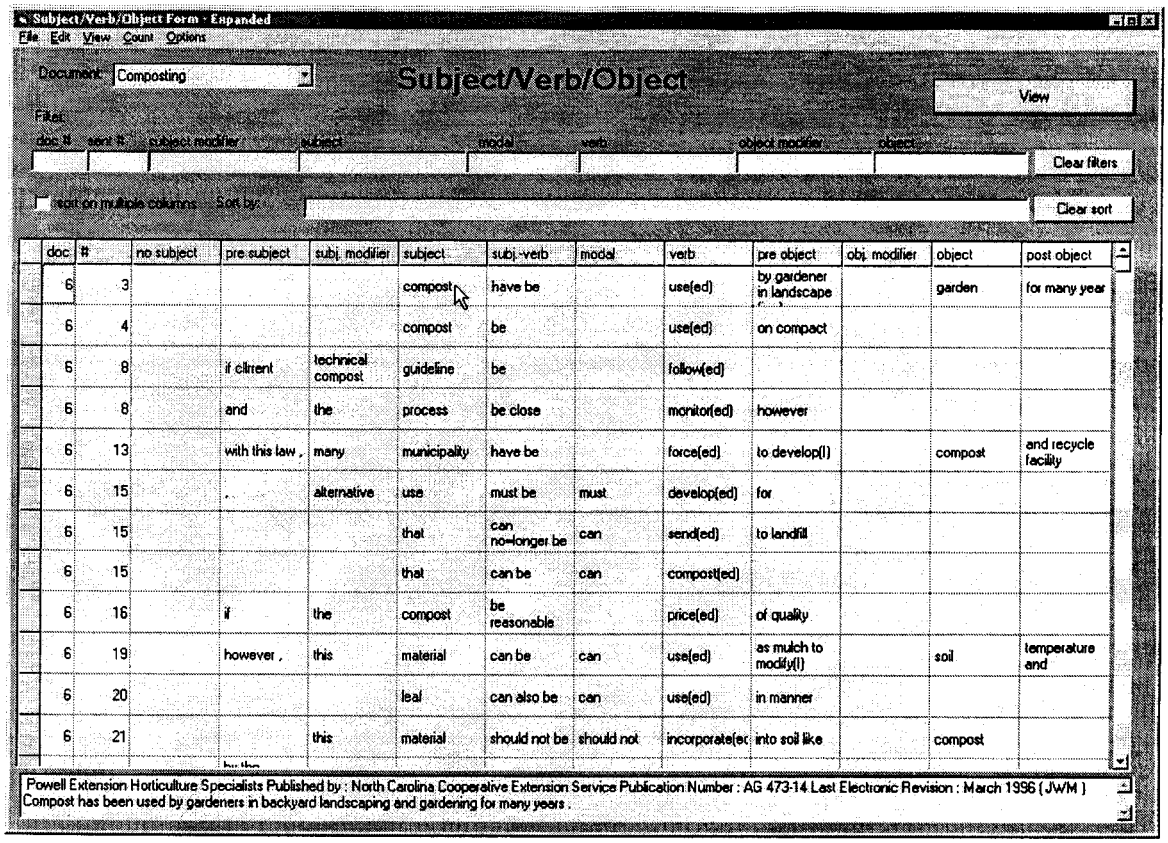



*Figure 10: Subject/Verb/Object Occurrences using TA*

| Type | Question |
|---|---|
| 1 | What is the definition of ___? |
| 2 | What is a broader term (super) for ___? |
| 3 | What kinds of ___ are there? |
| 4 | What parts does ___ have? |
| 5 | What is ___ a part of? |
| 6 | How do I ___ a ___? |
| 7 | What is the function of ___? |
| 8 | When does ___ ___? *or* When can I ___ ___? |

| Type | Question |
|---|---|
| 9a | What happens if ___? *or* What does ___ cause? |
| 9b | What causes ___? |
| 10 | What does ___ require/support? |
| 11a | Where is ___? |
| 11b | Where does ___? |
| *NA*[39] | What is the acronym for ___? |
| *NA* | What does the acronym ___ stand for? |
| *NA* | Tell me about ___ and ___. |

*Table 7: Question Types Answered by the TA*

[39] The questions related to acronyms are answered directly, using a database of acronym definitions. The question *Tell me about ___ and ___* simply runs a concordance. Unlike the other question types, these do not use the pattern-based answer extraction technique to derive answers.

The TA presents the question types to the user as templates where the user must fill in the blanks (Figure 11). The TA then searches through the document base to identify the relevant sentences and rank them according to the kinds of patterns they contain. The TA also performs limited anaphora resolution[40]. The output is shown as a concordance around some search word, with the answer extracted into a separate column (Figure 12). All displays are shown using grids based on database technology.

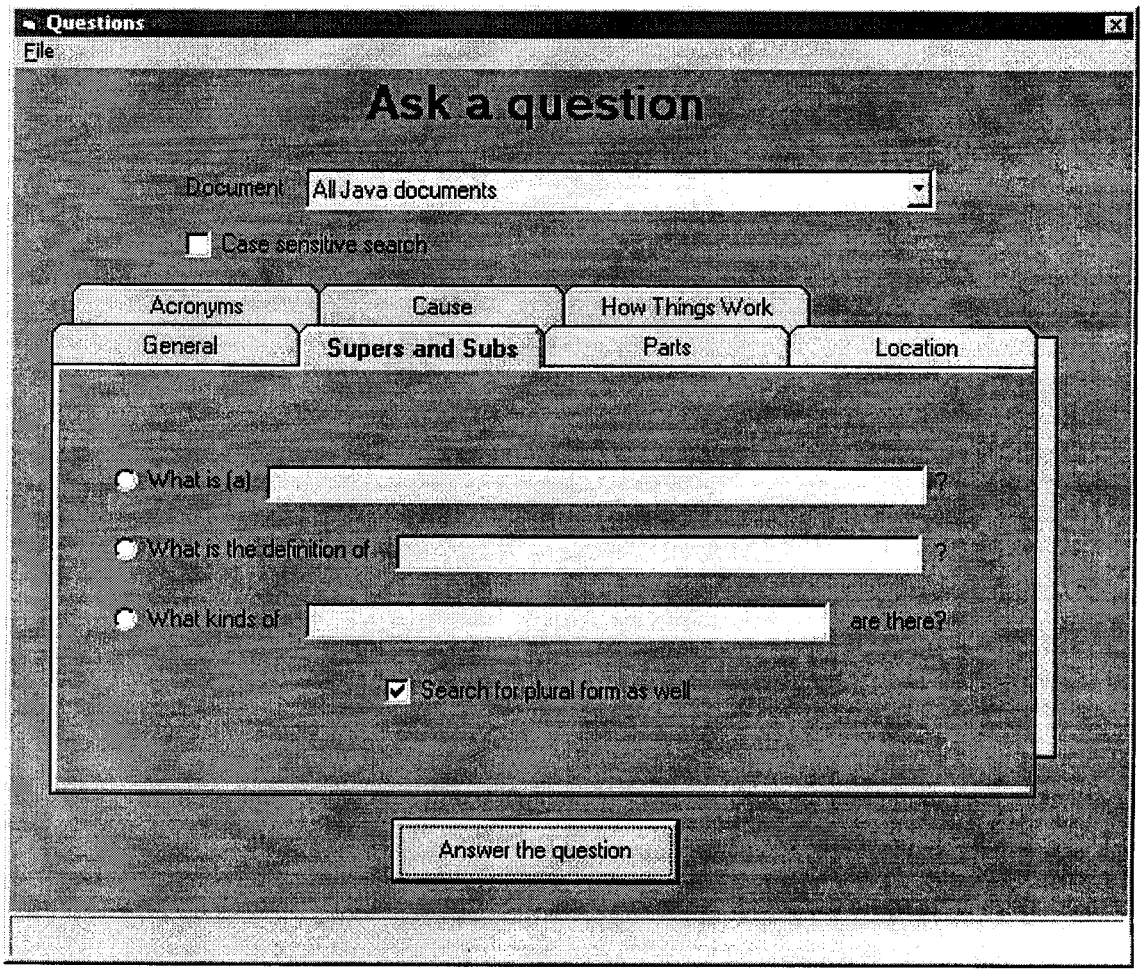



*Figure 11: Asking a Question using TA*

[40] Anaphora resolution implies determining the target being referenced by linguistic elements such as pronouns. The TA uses simple heuristics, for example, matching the pronoun *it* with the first preceding noun.

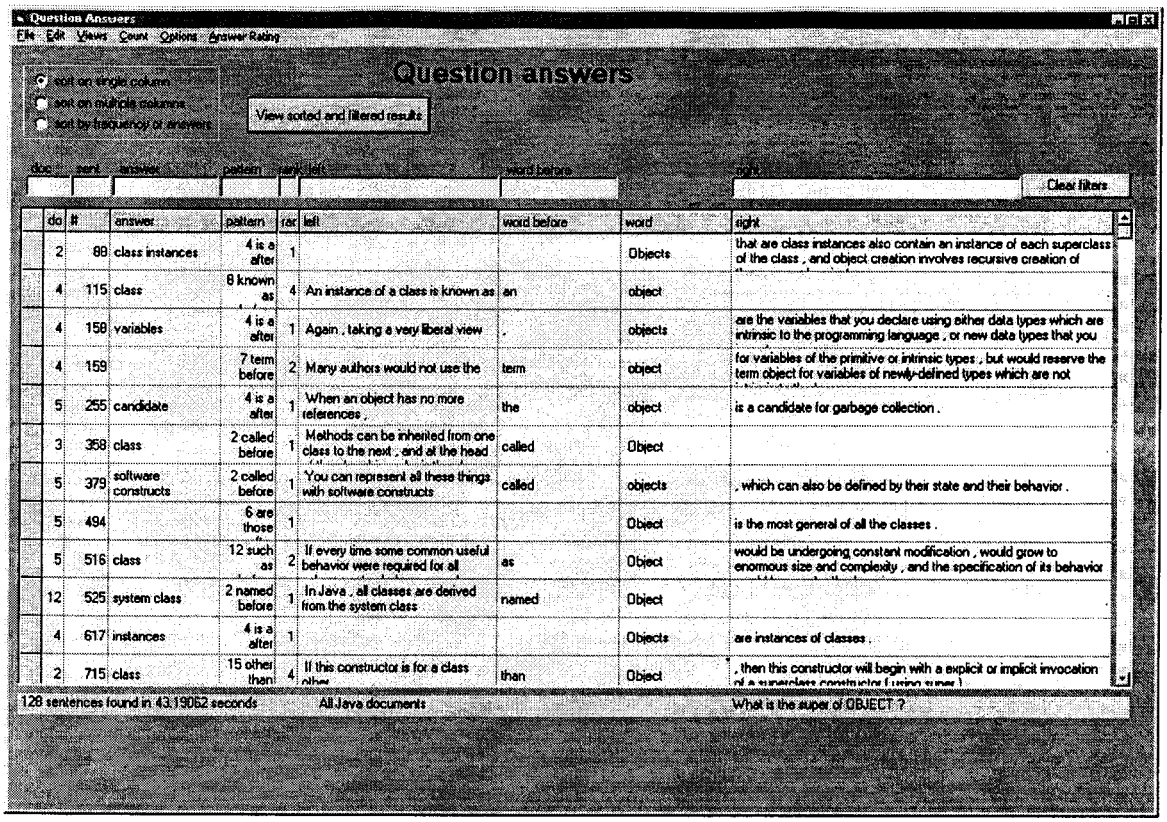



*Figure 12: Answers from TA*

### 3.2.1 Preprocessing

In order to enable the kinds of operation discussed so far, the TA first pre-processes the documents. The source input can be plain ASCII text files or HTML files, with a preference for the latter, as these support explicit paragraph divisions and may describe document structure in the form of nested levels of headings.

In the first stage of preprocessing, the files are filtered to delimit sentences and remove any HTML tags while maintaining information about paragraph divisions, lists, headers, and sections. The second stage applies LingSoft's ENGCG tagger [Voutilainen & Silvonen, 1996] to introduce part of speech and syntactic tags. The third stage simplifies the ENGCG tags, identifies embedded linguistic patterns and potential question types, and finally generates three tables for import into a database: a noun table with all noun occurrences, a verb table with all verb occurrences, and a sentences table with the structure described in Table 8. Figure 13 shows an example sentence with its tagged form, as visualized in the *Sentence Information* window of the TA. A fourth stage derives

additional summary statistics from the generated tables, and generates a list of potential compound words by finding all contiguous non-overlapping sequences of two, three, and four word noun co-occurrences, ordered by frequency.

| Column | Description |
|---|---|
| DocumentNumber | Unique document identifier. |
| ParagraphNumber | Paragraph sequence number within a document. |
| SentenceNumber | Sentence sequence number within a document. |
| Header | Heading level of a sentence that is a section heading. |
| PreviousHeaderPosition | Sentence number of previous section heading, if any. |
| List | Indicates if the sentence is part of a header or a list. |
| ListHeaderPosition | Sentence number of the start of the list, if any. |
| Patterns | List of all linguistic patterns embedded in the sentence. |
| QuestionTypes | List of all possible question types answered by this sentence. |
| Sentence | The sentence itself, as text. |
| TaggedSentence | The sentence itself, with tagging information. |

*Table 8: TA Sentence Table Structure*

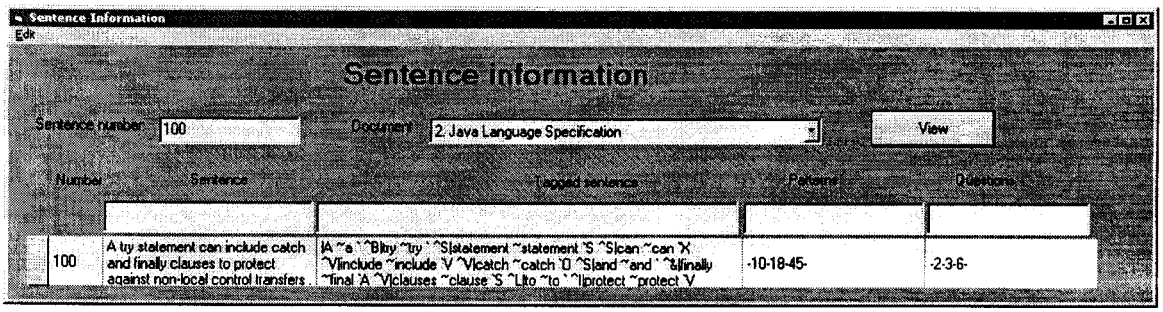



*Figure 13: Sentence Information in TA, having the Structure Presented in Table 8*

### 3.2.2 Indexing

At present, the TA does not employ any full-text indexing[41] of the sentences. Consequently, certain operations where retrieval of candidate sentences constitutes the greatest factor in processing time, such as question answering, incur a substantial albeit tolerable delay of about 30 to 40 seconds.

The LAKE group chose to avoid full-text indexing from its own practical research considerations and human resource limitations, following a series of performance evaluations using Oracle's ConText Cartridge full-text indexing engine. Table 9 shows

[41] Very recently, the LAKE group has begun using Glimpse [Manber & Wu, 1994] with the Text Analyzer.

the retrieval time of sentences using single word queries, with and without ConText, as applied to the group's research corpora[42]. Most queries showed a speed improvement having a factor of about 20 with indexing, which was somewhat less than the amount the group was hoping to achieve.

| | Retrieval Time (seconds) | | | |
|---|---|---|---|---|
| **Search Word** | **Sequential Scan** | **Indexed using ConText** | **Speed Factor** | |
| check | 31.40 | 9.25 | 3.4 | |
| host | 34.30 | 8.14 | 4.2 | |
| port | 31.81 | 2.80 | 11.4 | |
| keyboard | 27.87 | 1.44 | 19.4 | |
| disk | 27.97 | 1.18 | 23.7 | |
| subdirectory | 24.61 | 0.93 | 26.5 | |
| modification | 26.76 | 1.50 | 17.8 | |
| sort | 28.67 | 1.95 | 14.7 | |
| map | 23.77 | 1.33 | 17.9 | |
| compilation | 30.07 | 2.61 | 11.5 | |
| help | 31.67 | 9.43 | 3.36 | |
| current | 39.30 | 22.29 | 1.8 | |
| stream | 33.93 | 14.34 | 2.4 | |
| application | 33.15 | 30.00 | **1.1** | ← Top 12th term by frequency |
| Java | 34.19 | 125.32 | **0.27** | ← Top 5th term by frequency |

*Table 9: Retrieval Time With and Without Indexing (Courtesy John Talbot)*

Worse, the group found degenerate cases where retrieval time actually *increased* significantly. The group revealed these cases to occur on high frequency terms, present in most of the sentences of the group's research corpora. This revealed a performance penalty from indexing overhead. This is because high frequency words pose a challenge to indexing technology[43]; Context is not a state-of-the-art information retrieval system. Unfortunately, much of the group's experimentation focuses on knowledge engineering

[42] The research corpora included various documents related to the Java programming language. Its estimated size was about 40 megabytes.

[43] Retrieval time, such as from secondary storage, includes at least two components: the latency (such as disk seek time) and the transfer time. For very small units of retrieval, such as sentences, the former overwhelms the latter. Thus, a single full-text sequential scan, with reduced latency overhead, may provide better performance than numerous random accesses.

and acquisition. For the group, these high frequency words are of great importance, being the focus of analysis.

The group made some unsuccessful attempts at performance tuning. Given the difficulty of this process, and the administrative and programmatic complexity of using such indexing technology, the group decided to proceed without full-text indexing and to accept the delays. For document bases of up to 40 megabytes, such as the group's research corpora, the group found that indexing was not worth the trouble for its purposes.

### 3.2.3 Text Analyzer Server Mode

The TA offers a subset of its functionality for use by external systems. Its document base and summary data are stored in an Oracle database, which may be accessed directly. Beyond the interface available through the database schema, the TA defines a simple socket-based protocol for communication with the question answering operations. Through this protocol, an outside application may send requests to answer a given question, such as "`supers object`" and receive notification upon the completion of the operation. The results are stored in a database table, to be accessed by the external application. At present, the TA implementation has limited multi-user abilities and, without starting several instances of the TA application, cannot concurrently service multiple question answering requests from external systems.

## 3.3 The Knowledge Organizer (KO)

### 3.3.1 Overview

The Knowledge Organizer borrows heavily from its predecessor, DKBB. Figure 14 shows the main user interface, which resembles that of DKBB. It includes the hierarchy window, with an outline view of subjects, and grids that display the associated statements and facets.

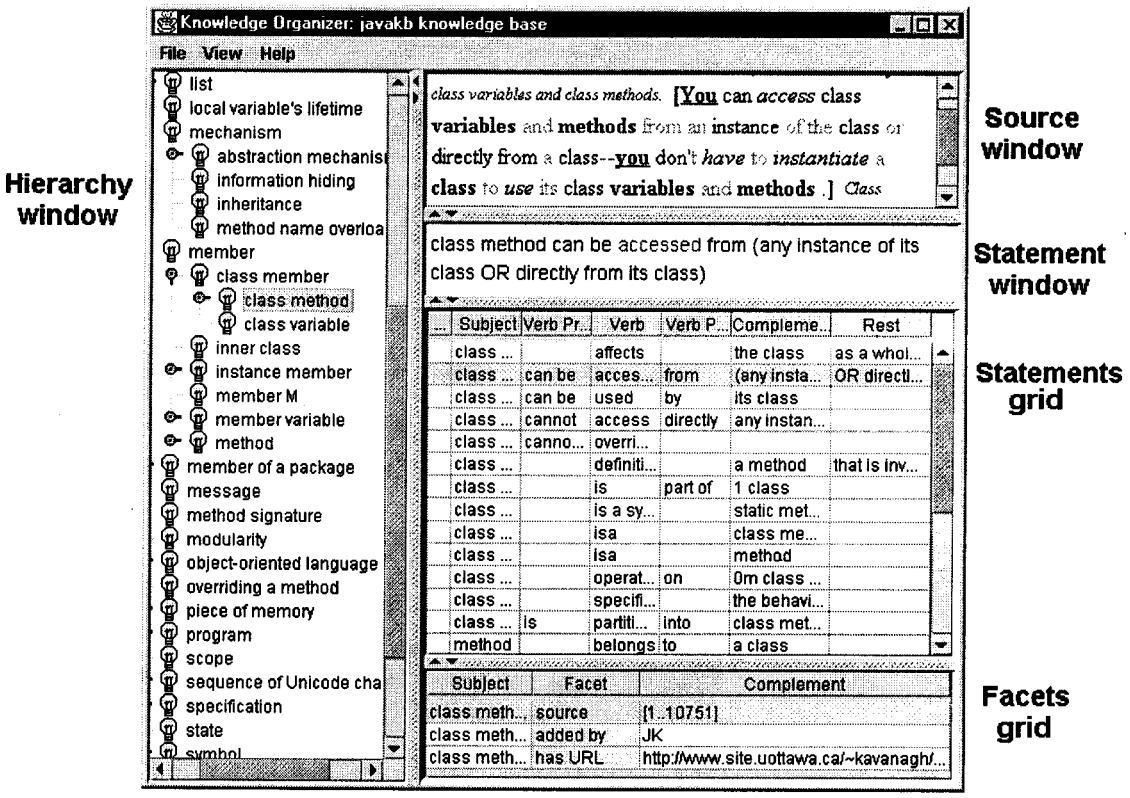



*Figure 14: The Knowledge Organizer (Advanced User Mode)*

Improvements in the user interface allow editing of statements to occur in-place, without the need for a separate statement entry area. The statement window presents the current statement as natural language text, for increased readability. The source window, borrowed from TAKODO, provides immediate access to passages from the document base associated with the knowledge base, as specified by URLs in statement facets.

The hierarchy window offers several visualization options:

- *Subjects by level* shows all subjects as an outline, just as DKBB. It is possible to choose among the type of relationship to be represented, such as hyponymy or meronymy. A subject that has the same such relationship with several subjects, such as a subject with multiple hypernyms, will appear several times in the outline, in all places that are relevant.

- *Subjects by level with statements* is similar to *Subjects by level*, but includes the statements corresponding to a subject as child nodes in the outline, with further child nodes for the facets. Different icons easily distinguish the subjects, statements, and facets visually.

- *Subjects alphabetically* is a top-level lexicographic list of all subjects. Each top-level node is the origin of an expandable outline view, as in *Subjects by level*, presenting child nodes according to the chosen type of relationship, such as hyponymy.

- *Subjects alphabetically with statements* is similar to *Subjects alphabetically*, but the display includes statements and facets, as in *Subjects by level with statements*.

The statements grid and facets grid employ color-coding to emphasize features such as non-inheriting predicates of facet-names. Furthermore, statements can be shown as complete sentences of natural language text (e.g. *a car is a vehicle*) rather than as distinct columns for the various parts of the statement (e.g. *car* | *isa* | *vehicle*).

The various windows may be resized and collapsed, as needed. In fact, the interface shown is that of the advanced user mode. The novice user can hide the source window and facets grid.

Borrowing from TAKODO once again, all grid views in KO may be resized, rearranged, sorted and filtered in a variety of ways, with columns selectively hidden or shown (for a description of these features, please consult the next chapter). This is

especially useful for the *All Statements View*, which is a separate window containing all statements meeting some criterion, regardless of their associated subject.

The graphical view of DKBB was integrated into KO. It remains essentially unchanged, preserving all visualization and knowledge manipulation functions.

The LAKE group made several significant changes to the knowledge management core, as compared to DKBB:

- As well as the subject, predicate, and complement, statements now include the *verb*, the *verb prefix*, the *verb postfix*, and the *rest*. The first three additions currently exist for presentation purposes, allowing the user to include conjugated verb forms, auxiliaries (to be or to have: *is*, *are*, *has*, *have*) or modifiers (adverbs, modals such as *can*, *should*), and prepositions (distinguishing *pick up* from *pick on*) to improve the display of statements as natural language; in the future, these additions may be used for other purposes, such as for inferencing. The concept of a complement in DKBB is replaced by the KO complement and rest. The former contains the phrase most closely associated with the verb. By convention, it should not include more than one noun phrase or adjective phrase or non-finite verb phrase (infinitive or gerund), but may include quantifiers and conjunctions (such as *and* and *or*). There are few conventions for the rest, which contains any other information, such as a phrase that modifies the complement or verb.

- Although lexemes still use a default sense number, if none is explicitly given, they may now specifically omit the sense number. This is different than using a default sense number, as it records the fact that the specific intended sense was unknown at the time of entry.

- Complements and the rest may contain HTML content, which is rendered within the KO interface. This allows the knowledge base to embed images and include links to other, external documents.

- The system allows compound statements, which may also be used to enter procedural information. Compound statements are formed by linking several

statements together, in sequence, to form a chain. Each link specifies the kind of relationship. For example, the compound statement "*if a vehicle has two wheels and it has one seat then it is a motorcycle*" could be entered as three statements related by the keywords *if*, *and*, *then* to form a compound statement. Whenever a statement is selected, the statement window shows it in the greater context of any compound statement of which it may be a part. At present, the compound statements do not have any special interpretations by the system. The inferencing engine does not currently apply any special logic for handling these compound statements, but this could be added in a subsequent version, using conventional AI technology.

- The system supports any number of hierarchies, including hyponymic and meronymic taxonomies. The hierarchy window allows the user to choose among these.

- All knowledge base elements (subjects, predicates, statements, etc.) maintain auditing information recording who and when a user created the element, and who and when it was last modified.

### 3.3.2 Implementation

Just as DKBB, The Knowledge Organizer is a Java-based applet suitable for deployment across the Web. It may also be executed as an application, outside of any Web browser. The current implementation targets the Java 2 Platform.

Altogether, KO consists of about 250 classes totaling over 1.4 megabytes of source code, not including portions borrowed from TAKODO. Approximately half of the system was designed and coded by the author of this thesis, and the other half by John Talbot. Additional input by other members of the LAKE group guided further development.

The architectural design separates the user interface from the knowledge management core through an abstract API. As a result, the core may be substituted at any

time with alternative implementations. The abstract API also defines the knowledge server interface to external applications, such as TAKODO.

The implementation adheres to principles of good object-oriented design [Gamma et al., 1995; Booch, 1993]. Knowledge base elements were factored into distinct classes and abstract interfaces, and were related into a class hierarchy. The class diagram in Figure 43 of the Appendix shows the abstract interfaces for knowledge base elements, and Figure 44 shows one implementation of these, as used for transient data. The server API is further depicted in Figure 45. Use of KO as a knowledge server does not require familiarity with any other, implementation-specific classes, such as those constituting the concrete implementation found in Figure 46.

The KO server performs central management, maintaining integrity constraints of the knowledge base. The underlying persistent storage mechanism uses the standard JDBC API for database interaction, and is essentially database-independent[44]. Retrieval of information from the persistent store is demand-based and accommodates very large knowledge bases, limited in theory[45] only by the amount of available secondary storage.

Caching of frequently used objects, using an exponential back-off model for cache cleanup, increases performance and provides greater responsiveness to the system. Prefetching optimizations, particularly at startup, but also available on request as specialized query hints, reduce the frequency of frame faults. For example, in the *Subjects alphabetically* display, all subjects, to some specified maximum threshold, may be prefetched in a single bulk operation. This would minimize the total delays from latency that might otherwise be incurred from retrieval performed one subject at a time. Given the multithreaded design, prefetching may be executed concurrently as an idle process.

---

[44] The Knowledge Organizer has been used with Oracle Corporation's Oracle RDBMS, Symantec's Sybase RDBMS, and Microsoft's Access RDBMS. There is still some divergence in the various vendors' implementations of the Structured Query Language (SQL).

[45] In practice, there are restrictions imposed by the time and amount of memory required to perform some kinds of computation, such as inferencing. Also, the user interface may limit how much information can be displayed simultaneously, such as in the graphical view of the conceptual taxonomy or of the semantic network. In this case, it is possible to restrict exploration of the knowledge base to some smaller subset, such as some given branch of the conceptual taxonomy.

Partial lexicographic and hashing indexes on the cache are maintained for very fast processing of several types of access that are used frequently, for example, retrieval by exact string matching of lexemes. The indexing mechanism maintains information about its degree of completeness. As such, in the case of a failure to retrieve information from the cache, it is possible to distinguish between nonexistent information and uncached information. Only the latter case will result in a frame fault requiring retrieval from the underlying database management system.

# Chapter 4: TAKODO

Several design goals were of primary importance for the development of TAKODO. The intent was to provide a useful tool for information retrieval and knowledge management. By useful, it was meant that the tool should provide a simple and efficient user interface, allowing rapid knowledge retrieval from diverse on-line sources. Furthermore, the author wished to design a flexible architecture enabling future extensions to the system with minimal impact.

As further motivation, a top ten list of research issues for applications of information retrieval, in the general sense, are presented in Table 10, along with a brief description of how TAKODO relates to each of these. The descriptions will be covered in greater depth throughout this and the next chapter.

With respect to knowledge management, the greatest impediments to knowledge sharing and reuse are presented in Table 11, with similar descriptions of their relevance to TAKODO.

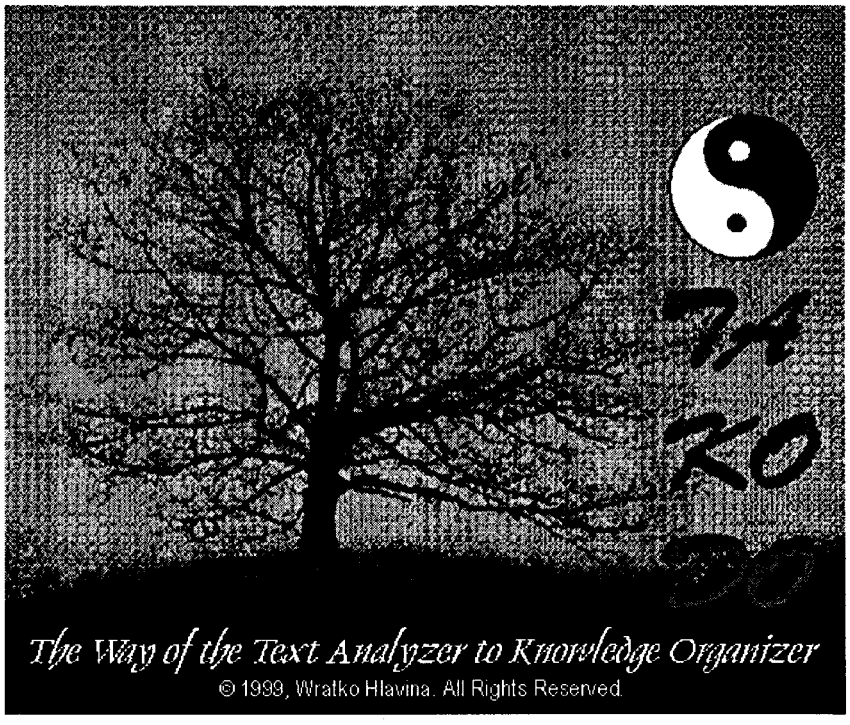



*Figure 15: TAKODO Splash Window*

| Research Issue in IR | Relation to TAKODO |
|---|---|
| 1. Integrated Solutions | Defines an abstract interface for knowledge sources. Combines information (passage) retrieval, corpus linguistics, and knowledge management using a common representation and visualization where possible. |
| 2. Distributed IR | *(Limited.)* Abstract interface, RMI, and multithreaded design allow communication with other information servers. Some support for collection fusion. |
| 3. Efficient, Flexible Indexing and Retrieval | Abstract interface for queries as executable objects, with support for efficiency optimization through a minimal specification of query hints. Retrieval modeled as multiple stages, in which indexing technology is easily replaced. |
| 4. "Magic" to address vocabulary mismatch | Automatic and manual query expansion. Knowledge-based query expansion dynamically driven by the current state of the knowledge base. |
| 5. Interfaces and Browsing | Simple and advanced user interfaces for query formulation. Simple presentation of results as a list, with numerous options for user-controlled sorting and filtering of retrieved query results. Visualization of query hits in context. |
| 6. Routing and Filtering | *Not applicable.* |
| 7. Effective Retrieval | Support for query expansion, stemming, restrictions on part-of-speech and syntactic tags; all of these allow human intervention. Passage retrieval reduces amount of reading needed by the user to find facts. |
| 8. Multimedia Retrieval | *Not applicable.* (Support for HTML allows embedding of media, but media is not directly searchable.) |
| 9. Information Extraction | Simple OAV modeling of clauses embedded in sentences. Text Analyzer performs answer extraction for common questions, provides conceptual operations to find hypernyms, meronyms, etc. |
| 10. Relevance Feedback | Results of retrieval can be saved to the knowledge base, with a relevance ranking. Changes to the knowledge base are immediately available for knowledge-based information retrieval operations. |

*Table 10: Top ten research issues for applications of IR [Croft, 1995]*

| Impediment to Knowledge Sharing | Relation to TAKODO |
|---|---|
| 1. Heterogeneous Representations | Defines an abstract interface for knowledge sources, as long as these can be modeled as OAV triplets. Dynamic, bi-directional translation between representations. |
| 2. Dialects within Language Families | (See above) |
| 3. Lack of Communication Conventions | (Limited; see above.) Does not currently use any official standards for communication of knowledge (KQML, KIF). |
| 4. Model Mismatches at the Knowledge Level (Lack of Shared Ontologies) | (Limited) Defines a minimal set of primitive relations, but otherwise does not impose any ontological commitments. Possible dynamic translation of statements using different terminology or ontologies (limited example found in the interface to DKBB). |

*Table 11: Impediments to Knowledge Sharing [Neches et al., 1991]*

## 4.1 Starting TAKODO

The TAKODO system can be started as either a Java applet or a Java application. As an applet, the system may be deployed across the Web using Java-enabled Web browsers. Alternatively, TAKODO may be launched as an application, independently of any browser.

## 4.2 Open TAKODO Wizard

On startup of the TAKODO system, the user is presented with the *Open TAKODO Wizard*. This wizard initiates connections between TAKODO and its various subsystems, requesting of the user any necessary startup information. The steps involved include, in order: login, connection to the knowledge base server and knowledge base selection, connection to the Text Analyzer server, and finally, selection of a document base.

## 4.3 Login and Audit Trails

In a context of multi-user knowledge management, there is a need to maintain some degree of security, to limit access to authorized individuals, as well as audit trails to identify who created or modified what knowledge. Subsequently, when information must be verified or reconciled with other, possibly contradictory, information, it will be possible to track the source of the knowledge, for further inquiries.

Therefore, initial interaction with the user, as the first step of the *Open TAKODO Wizard*, begins with a login process, requesting a user name and password (Figure 16). Note that the system allows the user to login anonymously as a guest, in which case direct access to the knowledge base may be severely restricted, but operation as an information retrieval tool remains fully operational. (In the future, there may be restrictions on which document bases can be viewed, in case the documents contain sensitive information.)

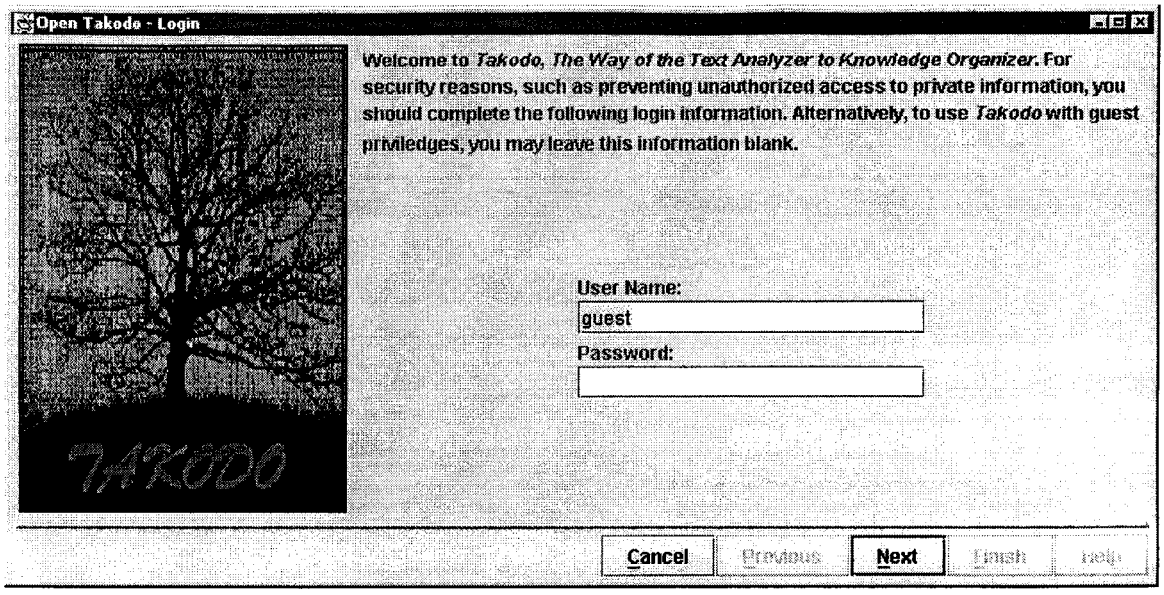



*Figure 16: Login to TAKODO*

## 4.4 Initial Selection of Information Sources

TAKODO includes its own knowledge management system, which is called TAKODO Candidates, and which manages a special knowledge base, which is called the **TAKODO Candidates Knowledge Base (CKB)**. This knowledge base serves several diverse needs, as will be described later. It is usually used as a temporary scratch pad, to store statements as candidates for subsequent entry into a final knowledge base.

In addition, TAKODO can connect to a main knowledge base and to an associated document base. Henceforth, all references to a **knowledge base (kb)** shall imply the main knowledge base, *excluding* the CKB unless otherwise stated.

A single knowledge base and a single document base are not sufficient to address the many, varied domains in which users may be interested. Thus, following the login procedure, the user can choose among several existing knowledge bases and documents (or document bases), from which to perform retrieval and to which acquired knowledge may be saved (Figure 17). The union of the CKB, the knowledge base, and the document base will henceforth be called the **document-knowledge base (dkb)**, consistent with the terminology of Skuce [1999a]. The union of the dkb and any other information sources

accessible to TAKODO (such as other databases and network-accessible resources) will henceforth be called the **information base**.

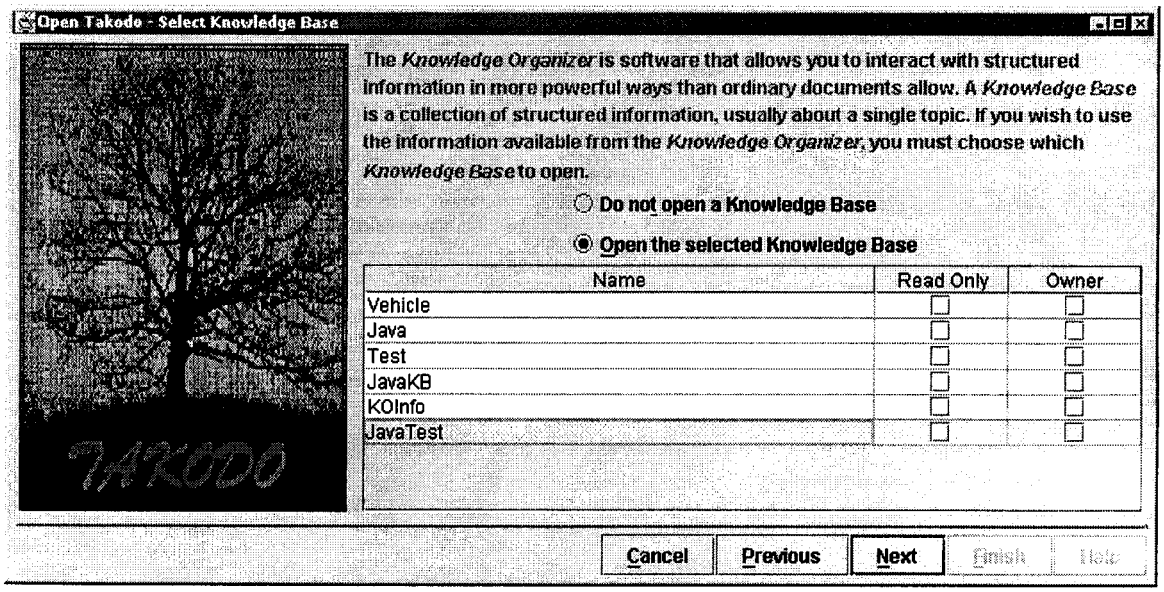



*Figure 17: Selection of a Knowledge Base in TAKODO*

Upon login, the list of available knowledge bases is retrieved by a background process, which may be aborted. Once obtained, the list is presented as a grid, indicating through several columns which are accessible as read only to the current user (disallowing modifications) and which are owned by the user. This display may be sorted and filtered, as with any grid display in TAKODO (sorting and filtering will be described later).

The use of a knowledge base is optional: if the user forgoes its selection, then operation of TAKODO as an information retrieval tool will continue, but without the benefit of such a knowledge base in knowledge-based query expansion (or any other features dependent on the existence of a knowledge base). Instead, operation will continue using only the Candidates Knowledge Base (CKB).

Similarly, the use of the Text Analyzer is optional. The system will attempt to automatically connect to the TA server following knowledge base selection. The user may abort this connection at any time, in which case operation of TAKODO will continue, but without the question answering features of the TA.

Each external knowledge base may be associated with its own document base (Figure 18). If the knowledge base has not already been associated with such a document base (or if no knowledge base was opened), then the final step before entering the main TAKODO screen will be the selection of a default document base (thenceforth associated with the knowledge base). The document base can be any set of available documents, through multiple selection of the items presented in the document list. Some items, such as "All Java Documents," represent a set of documents, that is, a document base, rather than a single document, allowing convenient access to common document bases. Once again, the interface allows sorting and filtering of the items.

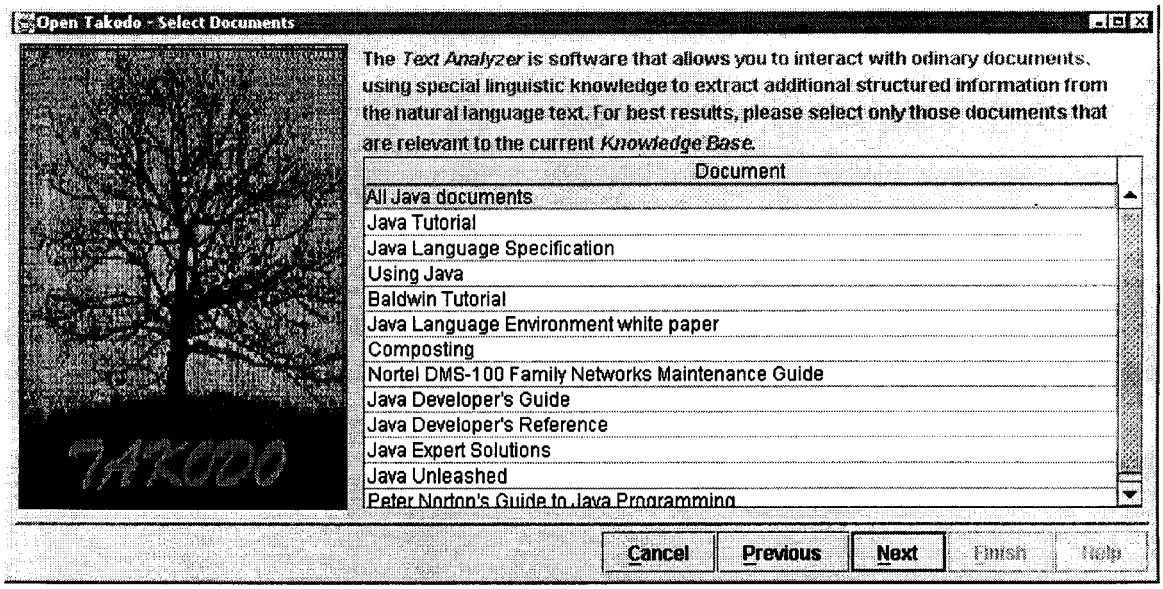



*Figure 18: Selection of a Document Base in TAKODO*

## 4.5 Main TAKODO Screen

The main TAKODO screen is presented in Figure 19. The user interface contains several areas, which will be described in order from top to bottom.

- The **menu bar** includes several menus: *System* provides access to development and administration functions. It is of limited use to most users, other than for importing and exporting knowledge between TAKODO and other external sources (for example, as part of a knowledge base archival procedure). *Edit* (Figure 20) provides access to knowledge base manipulation functions. *View*

(Figure 21) allows customization of the visualization aspects of TAKODO. *Find* (Figure 22) provides quick access to common questions, as well as options for specialized, advanced queries.

- The **toolbar** is a collection of icons for rapid access via mouse to very frequent operations. The toolbar may be placed in any of the four sides of the TAKODO window, and may be torn off into a separate window.
- The **query pane** allows the user to enter information needs in a simple representation. The query pane also provides a selection of available knowledge sources from which to perform retrieval.
- The **answers grid** provides the list of answers to the query, and is the main area of interest to the user. This grid has many features and visualization options, which will be covered separately.
- The ***Extras* pane** at the bottom provides access to diagnostics, additional information related to the current answers, and interfaces to other tools useful for information retrieval or knowledge management. The *messages pane* of this *Extras pane* logs the queries that were requested as well as any warnings or errors that were issued.
- The **status bar** shows current progress during retrieval, and summary statistics such as number of retrieved answers.

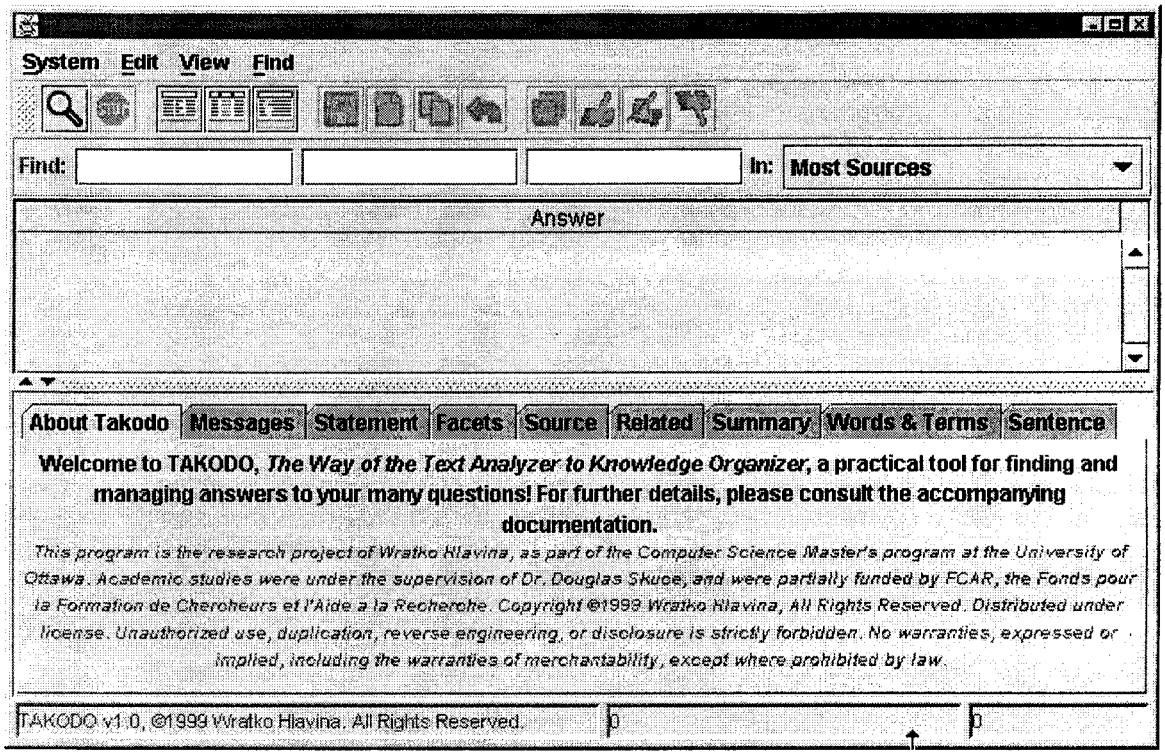



*Figure 19: Main TAKODO Screen*

The user interface[46] provides multiple ways for the user to access features. Beyond the menu bar and the toolbar, most components provide context-sensitive popup menus. Frequently used operations are available via keyboard shortcuts (accelerators) and global hotkeys. The remainder of this thesis will refer to features by name, where possible; details of how to invoke these features are deferred to a user manual.

[46] The toolbar, keyboard shortcuts and global hotkeys are configurable, via editing of an initialization file in a standard text format.

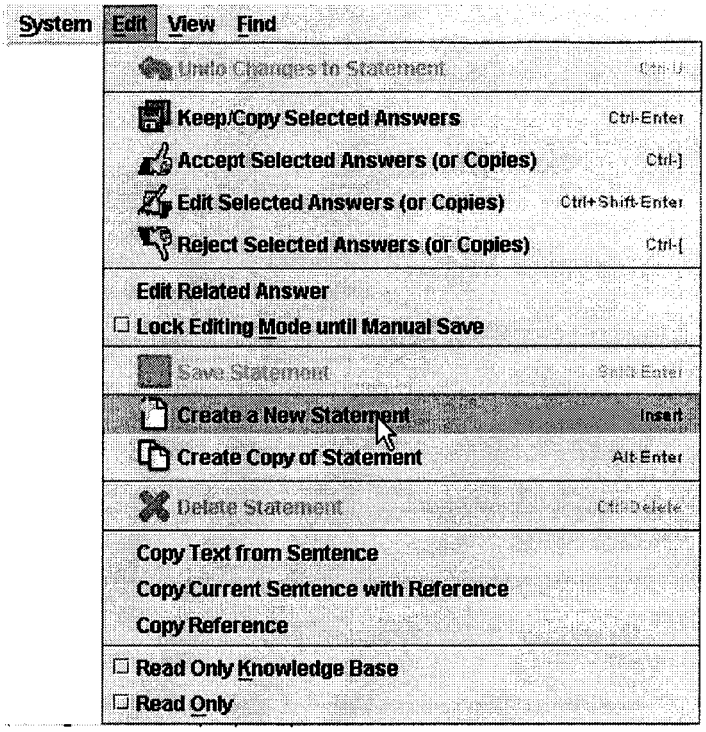



*Figure 20: Edit Menu of TAKODO*

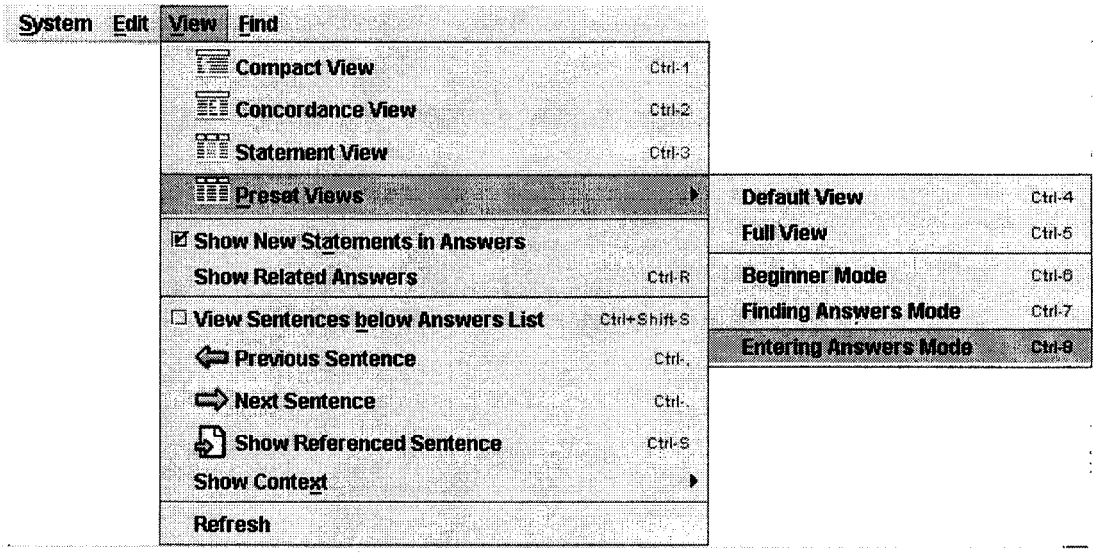



*Figure 21: View Menu of TAKODO*

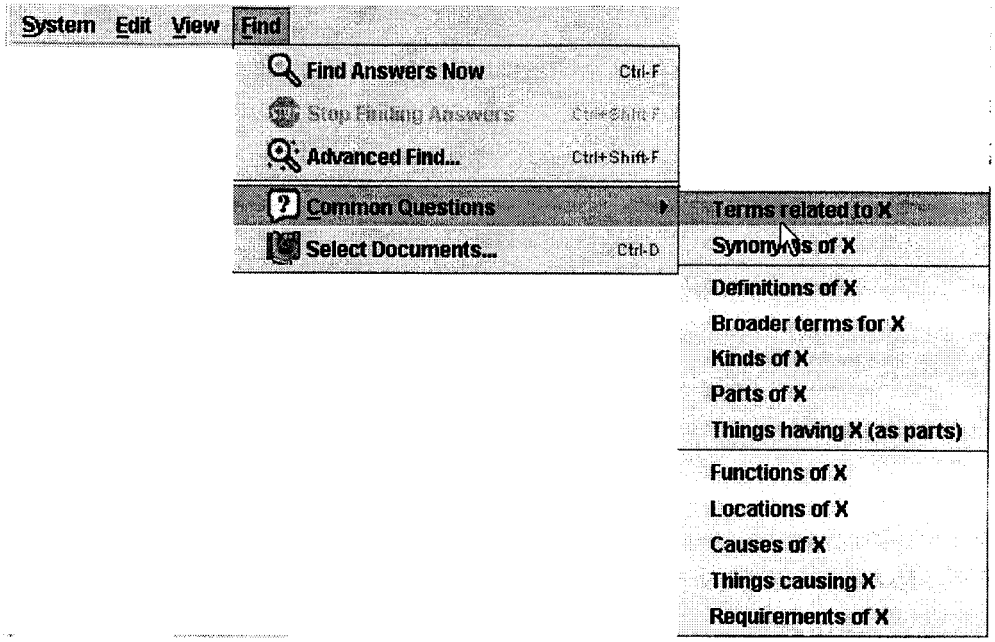



*Figure 22: Find Menu of TAKODO*

## 4.6 Knowledge Representation and Visualization

Before continuing with a presentation of the user interface, let us describe the underlying knowledge representation strategy.

In order to achieve an integrated solution that was simple (both programmatically and in terms of user interface), a common knowledge representation and visualization was desired. Minimally, the choice had to satisfy the constraints imposed by the inclusion of the Text Analyzer question answering system and frame-based knowledge management systems such as the Knowledge Organizer.

### 4.6.1 Statements and OAV Triplets

Note that the answers to a TA question can be represented as statements (subject-predicate-complement triplets, possibly with facets), which are essentially OAV triplets. Since the frames from the Knowledge Organizer can also be represented as OAV triplets, the latter became the final choice. Thus, knowledge in TAKODO is modeled, at its very least, as a set of OAV triplets.

### 4.6.2 Sentences

Common choices for document fragments include, for example, fragments motivated by the subtopic structure, as discussed previously, as well as simpler approaches such as individual paragraphs, or all possible windows of *n* sentences. Following the example of the TA, the basic unit of retrieval for the purpose of passage retrieval in TAKODO is the sentence. Justification for this choice lies in that (i) TAKODO's main purpose is closely related to question answering, (ii) most answers can be found in fragments of one or two sentences [Ambroziak & Woods, 1998], (iii) a definition[47] of a sentence is "a complete expression of a single thought" [Crystal, 1997], which represents the ideal during the writing process, and (iv) a larger context for the retrieved sentence may be displayed when needed, provided a link to the source document is maintained.

### 4.6.3 Clauses

We of course wish to relate full sentences to the chosen fundamental model of knowledge, as OAV triplets. Ideally, the information contained within sentences would be re-represented as formal knowledge base statements. This is an information extraction task, but one that lacks the constraint of having a single, specific application domain for which to design suitable templates. As such, a completely general solution is not possible at present [Kilgariff, 1997a], but TAKODO does try to approximate the content of the sentences, for the benefits of information retrieval and rapid, semi-automated knowledge entry.

Most sentences [Crystal, 1997] are in the form of *major sentences*, which are either *simple sentences* (a single clause) or *multiple sentences* (multiple clauses conjoined, as with a linking word). Hence, TAKODO transforms sentences into OAV triplets by identifying embedded clauses and mapping subject-verb-object triplets into OAV triplets. This technique was used in the START natural language system [Katz, 1997], which extracted *kernel sentences* (parts of sentences, usually containing one verb)

[47] This is the old, notional definition of a sentence, rather than the more formal definition. The notional definition is inaccurate, as sentences may actually contain several complete thoughts, and some complete thoughts are expressed by fragments of sentences. [Crystal, 1997]

for entry into a knowledge base. There may be loss of some meaning in the translation, especially since this approach is less sophisticated, but it is felt that the ability to view the source sentence, in context, limits any disadvantages.

### 4.6.4 Answers

In TAKODO, the minimal result of retrieval is called an **answer**[48]. It is a row with several columns. Ideally, an answer should include, as one of its columns, a default representation for visualization purposes. Normally, a number of answers appear as a table using a grid display. TAKODO supports answers represented as plain text or as arbitrary HTML, including links to embedded media. Thus, an answer could be a graphic or audio file.

To match the OAV model, answers are expected[49] to have at least three columns, but may have others with additional information (Figure 23). The recommended statement columns match[50] those of KO and share the same semantics.

| Weight | Subject | Predic... | Complement | Left | Word | Right |
|---|---|---|---|---|---|---|
| | method | be | method | static indicates that the ... | method | is a class method. |
| | all, method, variable | be | method, variable | All of System's | methods | and variables are class methods and class va... |
| | method | be | method | The next | method | in the SimpleThread class is the run method. |
| | class | begin | definition | The keyword | class | begins the class definition for a class named ... |
| | class | be | high | The | class | is pretty high in the Throwable class hierarchy. |
| | class | be | similar | The CheckedInputStream | class | is very similar to the CheckedOutputStream cla... |

*Figure 23: Answers with Columns in TAKODO*

### 4.6.5 Beliefs

In practice, information is of varying quality. TAKODO does not assume all information available to it is 100% accurate and free of contradictions. Hence, information items in TAKODO are viewed as *beliefs*, with some certainty factor that a given fact is true. This certainty factor is represented as an aggregate of two values: the *status* and the *weight*. The status, like Cyc's truth values [Lenat & Guha, 1989], is measured on a five-point scale: *reject* (believed false), *reject candidate* (possibly false,

---

[48] As will be seen later, answers are, in actual fact, software components, implemented as objects (in the Object-Oriented Programming sense) with some restrictions.

[49] Answers are not required to have parts mapping to the subject, predicate, and complement OAV triplet. If these or any other parts are missing, the corresponding parts will be assumed empty.

[50] Support for KO's *verb*, *verb-prefix*, and *verb-postfix* have been omitted in the current version of TAKODO.

but not sure), *default* (no information on truth value), *accept candidate* (possibly true, but not sure), and *accept* (believed true). The weight is a variable scale, from 0% to 100%.

The rationale for having beliefs with two scales is that the fine granularity of the latter and its semantics may be confusing to users wishing to assign certainty factors [Lenat & Guha, 1989]. They may exhibit difficulty understanding the subtle difference and choosing between, say, a 20% and a 30% certainty factor. However, this distinction remains useful for automated processing or for relevance ranking.

### 4.6.6 Linguistic Issues, Ontology, and Representational Conventions

Similar to their use in KO, subjects, predicates, complements, and facets in TAKODO are lexemes, composed of a textual name and a sense number for sense disambiguation. The notation is also similar: `name^sense`. The sense number may be omitted, which is represented as `name^?`. Such a lexeme will match any sense of another lexeme having the same name. Furthermore, the notation `<name>` implies a system-level sense, usually meant for internal processing by TAKODO, rather than for user manipulation.

TAKODO does not impose any specific ontology. However, in order to allow sharing of information between the various subsystems and knowledge bases, meta-data must correspond, and a minimal amount of fixed semantics is necessary [Skuce, 1995b; Skuce, 1999a; Lenat, 1995]. Therefore, TAKODO defines a minimum number of primitive concepts and predicates (Table 12) as well as several columns (Table 13) with preassigned semantics.

| Lexemes | | Description |
|---|---|---|
| **Subjects (Concepts)** | <KBTOP> | Top of the concept hierarchy. |
| | <UNPARENTED> | Concept meant to group all other concepts whose position in the concept hierarchy has not been established. |
| **Predicates** | isa, sub | Generic relationships (BTG and NTG thesaural relationships; TA Question Types #2 and #3). |
| | topic | Topic/Subtopic relationships. |
| | has part, part of | Partitive relationships (BTP and NTP thesaural relationships; TA Question Types #4 and #5). |
| | term | PT thesaural relationships. |
| | synonym | SYN thesaural relationships. |
| | definition | Definition of the concept. (TA Question Type #1) |
| | function | Function or purpose (TA Question Type #7) |
| | location | Physical location (TA Question Type #11) |
| | causes, caused by | Causality relationships (TA Question Type #9) |
| | requires | Requirement relationship (TA Question Type #10) |
| **Facets** | rest | Statement's rest. |
| | source | Statement's source, which is normally a URL. |

*Table 12: Predefined Lexemes in TAKODO*

Unlike KO, hierarchical relationships are also represented as statements using the uniform OAV model. These statements use system predicates, such as `<isa>` for the hyponymic taxonomy. TAKODO relaxes the constraints on the taxonomy, as compared to KO, allowing contradictions and cycles, and using certainty factors to record relationships that may be uncertain or false. Whenever a strict hierarchy is needed, such as during inferencing, the system will use all relationships that have a belief value of *Accept Candidate* or greater, and will avoid cycles by breaking them arbitrarily. In the future, TAKODO will provide diagnostic warnings, notifying the user of such problems with the hierarchy.

Of course, problems with the hyponymic taxonomy should be rectified by the user when the desired product is a proper knowledge base. To this effect, TAKODO offers some specialized facilities. At present, these are at an experimental stage, and will be described in the section on finalization of the taxonomy.

| Columns | | Description |
|---|---|---|
| **Statements (OAV Model)** | Subject | Subject of the statement, as a Lexeme. |
| | Predicate | Predicate of the statement, as a Lexeme. |
| | Complement | Complement of the statement, as a Lexeme. |
| | Rest | Rest of the complement of the statement, as text. |
| | Facets | Additional facets of the statement or its complement, as a map from a facet to the value. |
| **Audit Trails** | Sentence | Actual sentence of context from which the answer was derived. |
| | Left, Word, Right | Surrounding context, within the sentence, and around which the answer was derived. |
| | Source, Subsource | Textual description of the information source from which the answer was described, partitioned as a primary source and secondary information relating to that source. |
| | Document, Document URL; Reference (Document#, Paragraph#, Sentence#, Character#, Length) | Reference to the document/passage from which the answer was derived. The Document URL is a URL to the document; the reference is in the format[51] [doc#.par#.sent#]. Document#, Paragraph#, and Sentence# are unique integer identifiers, assigned by the TA. Character# is an integer character offset. |
| **Relevance Ranking** | Status | Accepted/Rejected status. |
| | Weight | Certainty factor (or relevance weight), on a real-valued scale from 0 (irrelevant) to 1 (relevant). |
| **Visualization** | Answer | Most appropriate representation for the answer, as text or HTML. |

*Table 13: Predefined Answer Columns in TAKODO (Non-Exhaustive List)*

### 4.6.7 Visualization as a Grid

Knowledge management tools, such as the Knowledge Organizer (KO), usually employ outline views and graph views to display their conceptual taxonomies. This is acceptable for small knowledge bases with limited taxonomies. In TAKODO, the knowledge base may contain a small amount of high quality knowledge, mixed with a large amount of lower-quality knowledge. For this, navigation becomes inefficient (as was noted for hypermedia and in early experiments with Cyc). Instead, knowledge in TAKODO is accessed via queries that present their information in a grid with several rows and columns. As conceptual taxonomies remain important, TAKODO offers a

[51] The exact specification of the Reference format is deferred to a user manual.

wealth of filtering and grouping options, some of which are specifically designed to operate based on hierarchies of various kinds.

## 4.7 Information Retrieval Features

Information retrieval in TAKODO consists of two phases: initial ad-hoc information retrieval, as defined in section 2.3, followed by subsequent filtering[52] and sorting. This two-phase retrieval borrows from the idea of partial indexing, as used in Glimpse [Manber & Wu, 1994] or the Web Information Collector (WIC) [Zayour, 1997], for example. The initial ad-hoc information retrieval phase accesses a very large information base, but is expected to provide a collection of retrieved items that is significantly smaller. Therefore, the second phase offers more complex query operations than those allowed by the efficiency constraints of the initial phase. Final sorting complements any prior relevance ranking, and allows, for instance, presentation of results according to alternate relevance ranking criteria, or as part of collection fusion.

### 4.7.1 Concurrency

In an interactive system, response time is important. The user should not wait for retrieval of all relevant items before proceeding with their evaluation. This is especially true when items are relevance ranked, wherein the first few items are more likely to satisfy the user's information needs; the remaining items then become redundant.

Thus, in TAKODO, retrieval is performed concurrently with browsing of the results. As items are found relevant, they are appended to the results lists, without impeding the user in whatever other tasks exist.

At any time, the user may abort the concurrent retrieval operation, such as when the information needs are met by the items retrieved so far.

---

[52] The term *filtering* is used here in its general sense, as any process that accepts or rejects candidate items. The reader should avoid confusing this with *information filtering*, as described previously with respect to the field of IR.

### 4.7.2 Intended Query Semantics

Most IR systems employ one or more of several common query interfaces:

- Term List: A single text entry field allows the user to type in a list of query terms (with possible term truncation and phrase retrieval, among other features). The term list may be interpreted as meaning a search for the conjunction of all terms, the disjunction, or as a term vector for purposes of similarity computation, possibly depending on some option that is set in the query interface. A term list has the advantages of simplicity and rapid entry.
- Natural Language Query: A single text entry field allows the user to type in a fully formed question in natural language, such as in English. Often, a natural language query is interpreted only as a simple term list, with a stoplist for the function words commonly used to express questions.
- Boolean Query Expressed as a Textual Formula: A text field allows the user to type in a Boolean logic specification of the query. Although such an interface requires significant familiarity and expertise for effective use, studies comparing it with a natural language query interface are inconclusive, showing both achieve similar retrieval performance in terms of precision and recall [Hersh et al., 1999].
- Fill-in-the-Fields Query: Several entry fields, each with their own (usually fixed) semantics, allow the specification of restrictions on retrieved items. Examples of fields include Author, Date, and Title. These are therefore similar to database management systems.

For TAKODO, it was decided that the query interface should be modeled on OAV triplets, as they are used in the knowledge base, and likened to simple sentences of natural language. Thus, a query is composed of three parts: the intended[53] subject, the intended predicate, and the intended complement. Each of these three parts is presented, in its simple form, as a term list, with automatic query expansion. Hence, the interface can be

[53] We qualify the parts as *intended* since the nature of retrieval from a document base, containing ambiguous natural language texts, prevents identifying with absolute certainty whether a passage uses the words of the query with exactly the specified relationships, as subject, predicate, and complement.

viewed as a hybrid between the term list and a restricted natural language query. For example, to ask "What parts does an object have?" the user instead requests "`object/has part/`*blank*" (see Figure 24).

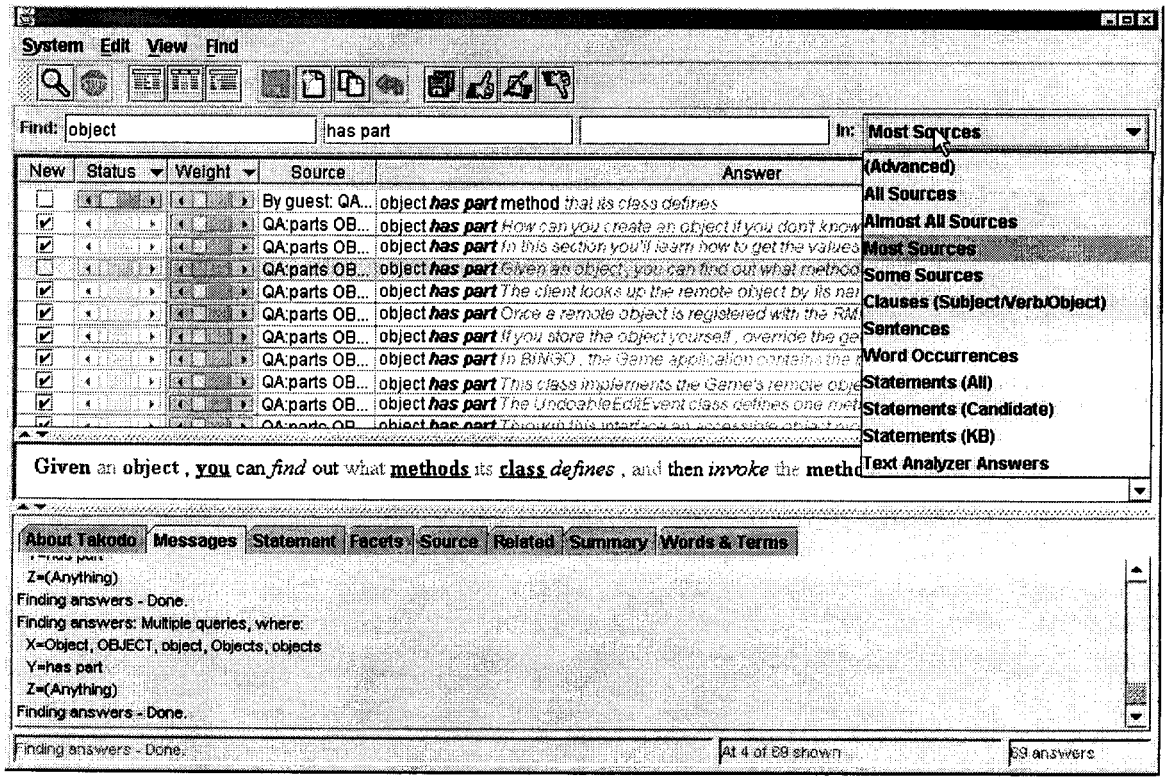



*Figure 24: Query Interface and Information Sources in TAKODO*

Feedback on query expansion, as will be described later, is in the form of popup hints, or ToolTips, that appear whenever the mouse hovers over a term list. Figure 25 shows the result of the default query expansion applied to the term list "`object, class, method`" when used on a document base dealing with the Java programming language.

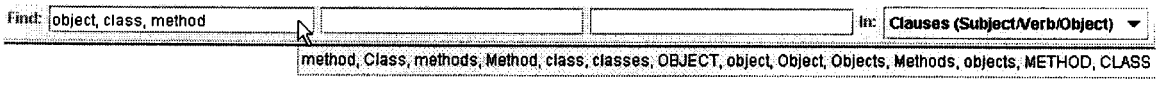



*Figure 25: Query Expansion Feedback in TAKODO*

The use of a fill-in-the-fields display provides clear semantics to the three parts, allowing the system to avoid the difficulties of parsing arbitrary, ambiguous natural language queries. It eliminates the need for typing function words (*What, does, an*) that serve a limited purpose for retrieval. It is further hoped that the interface acts as an

incentive to the user for expressing queries with greater clarity. Subsequent studies are needed to evaluate the benefits, if any, of this query interface.

As with many IR systems, TAKODO provides a simple query interface and an advanced query interface. The simple interface is the one just discussed. The advanced interface, discussed later, allows expressing additional constraints, some of which are outside the representational capabilities of the OAV query model.

### 4.7.3 Information Sources

So far, the discussion has focused on *what* to find, but it is also necessary to know *where* to search. Since TAKODO's information sources are diverse and of varying quality, TAKODO allows the user to selectively enable and disable information sources as part of the query. In the simple query interface, this selection is limited to some predefined sets (presets) of information sources, summarized in Table 14. The advanced query interface allows arbitrary selections other than those defined by the presets. Such a condition is indicated with `(Advanced)` shown in the simple query interface.

| Preset Sources | Description of the Information Sources |
|---|---|
| All Sources | All information sources are consulted. |
| Almost All Sources | All information sources, except word occurrences. |
| Most Sources | All information sources, except sentences and word occurrences. *(This is the default preset.)* |
| Some Sources | All statements and Text Analyzer definitions and supers (genus) |
| Clauses (Subject/Verb/Object) | Passage retrieval from the document base, defining the unit of retrieval to be the document fragments consisting of single clauses (as embedded in sentences). Subsumes sentences. |
| Sentences | Passage retrieval from the document base, defining the unit of retrieval to be the document fragments consisting of single sentences. |
| Word Occurrences | Passage retrieval from the document base, defining the unit of retrieval to be the document fragments consisting of single words (word-tokens) in context. Subsumes sentences. |
| Statements (All) | All statements from the CKB and the knowledge base. |
| Statements (Candidate) | All statements from the Candidates Knowledge Base (CKB) |
| Statements (KB) | All statements from the knowledge base. |
| Text Analyzer Answers | All answers derived by the TA from the document base. |

*Table 14: Information Sources (Presets) used by TAKODO*

### 4.7.4 Visualization Options

The visualization options for answers, which include column hiding and moving as well as row filtering and sorting, are quickly accessed from the context-sensitive menu associated with the column headers of the answers grid (Figure 26). In fact, these same options are consistently available in any other grid views within TAKODO, such as for presentation of term lists.

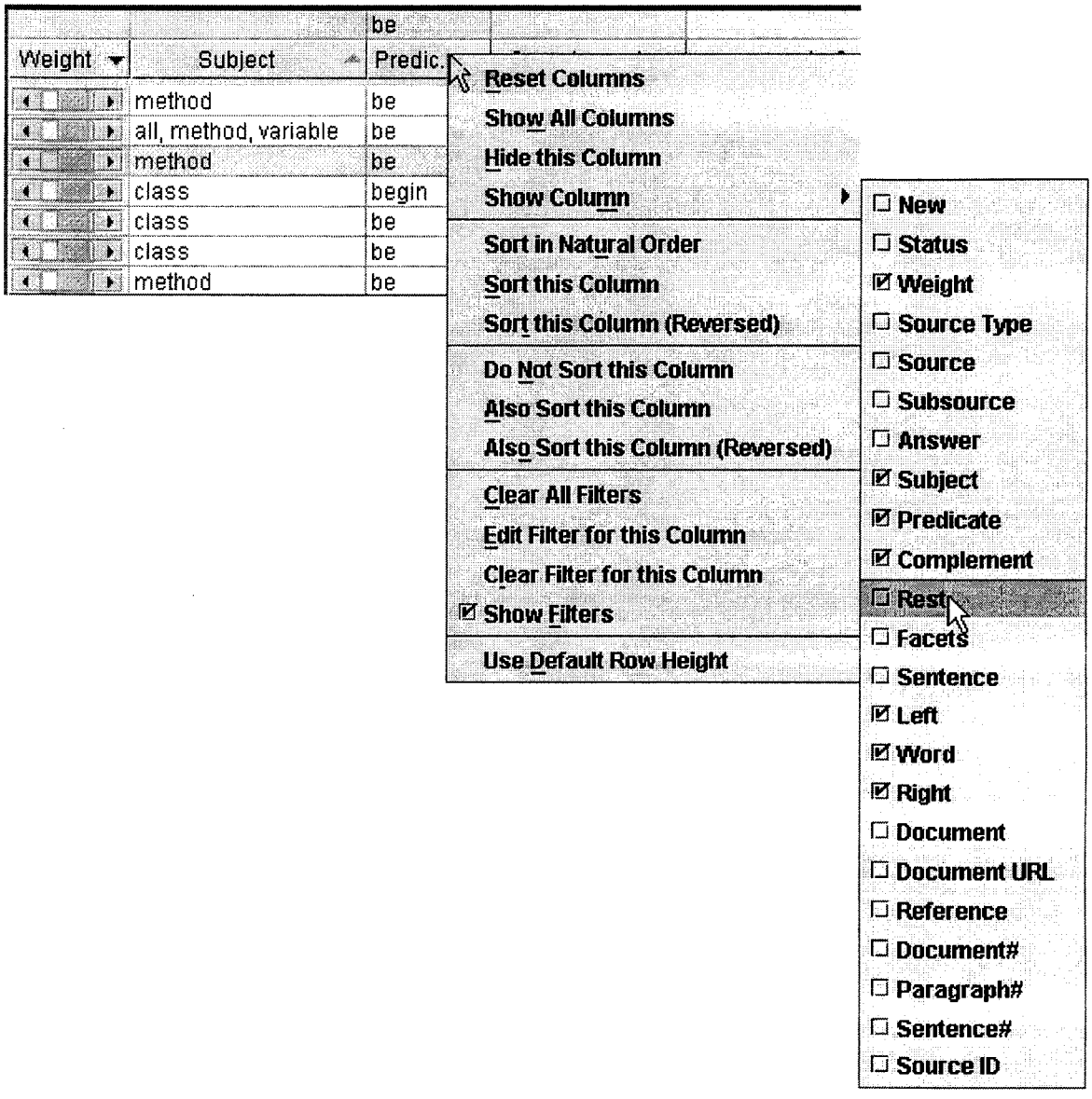



*Figure 26: Advanced Grid Visualization Options in TAKODO*

#### *4.7.4.1 Column Resizing, Rearranging and Hiding*

As described, answers may have many columns of associated information. These may be independently resized, and the columns rearranged to show more important columns first. Not all columns are necessarily of interest to the user; their display would simply obfuscate the other, more important information. For such cases, columns may be selectively and dynamically hidden or shown, as is most appropriate. The visibility status of a column merely affects the screen presentation. It does not affect the underlying data, nor does it limit operations on those columns, such as filtering or sorting. Indeed, it is possible to temporarily show a column, define a filter or sorting criteria on it, and then hide the column from further display while preserving the filter or sorting criteria.

In practice, several combinations of visible columns were found relevant for different tasks. These are rapidly accessed via icons on the toolbar and items under the *View* menu. Hence, it is very easy to jump between, for example, a statement-style display of sentence clauses with subject, verb, and object, to a concordance style KWIC display, or to a merged display as was shown in Figure 23. Users of TAKODO find these three most useful.

#### *4.7.4.2 Filtering*

Grid displays in TAKODO allow the user to specify additional constraints on the values of the various columns. These constraints define a visibility filter on the underlying data. By default, the filtering options are hidden in the user interface, in order to avoid confusing novices when the feature is not needed, but the options are quickly accessed from the context-sensitive menu associated with the column headers.

When visible, filters appear above the column headers. The syntax for filters is presented[54] in Table 17 within the appendix. It includes Boolean operators, string matching based on regular expressions, numeric comparisons, word classes, and thesaural restrictions. For example, the filter `object` in the Subject column would match any answer where the subject contains the specified word, without case sensitivity. The same

[54] The filter syntax may continue to undergo some revision. A complete specification of the syntax, in BNF notation, is not available at present.

filter, using `/^[Oo]bjects?`, would match any subject that starts with *object*, *Object*, *object*, or *Objects*, with case sensitivity. Again, using the hierarchy filter `@isa:vehicle`, it will match *vehicle* and all of its hyponyms, such as *car*. A filter of `@class:v/-` on a column whose contents contains tagging information, such as word occurrences (word-tokens), will find any word used as a verb and that is not part of the stoplist.

#### *4.7.4.3 Sorting*

Finally, grid rows may be sorted interactively and dynamically, on multiple columns. The sort order may be chosen as ascending or descending. Sorting is according to the natural collation order corresponding to a given column. The natural collation order is lexicographic for textual data and numeric for numbers. Lexemes are ordered first lexicographically by name, then numerically by sense number.

The user interface provides feedback on the current sorting order via arrow-shaped indicators on the columns, shaded to represent the sorting level (darkest for outermost sorting level to lightest for nested sorting levels).

Of course, rows may be left in their natural sequence, without any sorting. This does not imply the lack of some sorting order; for example, answers may be retrieved in order of relevance, in which case explicitly sorting by relevance weight is redundant[55].

#### *4.7.4.4 Browsing Hits in Context within Source Documents*

While viewing the retrieved items, the user may wish to refer to the source documents, perhaps to see the greater context of selected passages, or to verify the accuracy of any translation from the source text into a KB statement. The *Sentences* pane serves this role, displaying a scrollable view of the associated passage, with an additional amount of user-configurable context (*n* sentences, *n* paragraphs, etc.); the user may browse anywhere within the source document, or to other documents, moving the passage of interest where needed. This pane is located as part of the *Extras* pane, but may be torn off into the main window (Figure 24), given its importance.

---

[55] The sorting algorithm is optimized for this case.

The *Sentences* pane displays its content with various styles of formatting that highlight relevant passages, and bring attention to important elements, as hinted by the syntactic structure of a sentence. For example, subjects, verbs, and complements are emphasized. Linguists may inspect the tagging information associated with any word (Figure 27).

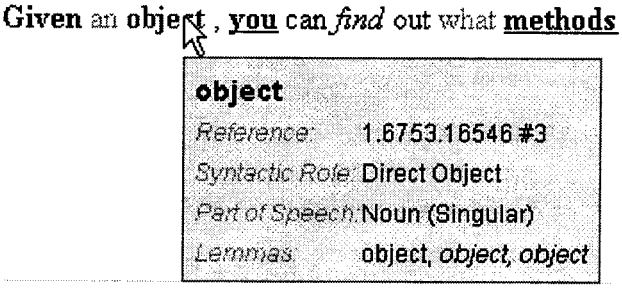



*Figure 27: Sentence Pane with Tagging Information in TAKODO*

### 4.7.5 Advanced Queries

The advanced query interface offers considerably greater expressiveness than the simple query interface. It is available as a dialog, with several tabbed panes: the options pane, and three other, identical panes, for each of the three OAV parts of the query model.

The options pane (Figure 28) shows the various information sources available to TAKODO, presented in a grid that may be sorted and filtered as desired. Each source may be selectively enabled and disabled, as previously discussed.

To the right of the information sources grid, one may see further query options, starting with the defaults to use for query expansion (Table 15). Any combination of query expansion options may be chosen. Below this, the user may specify limits on the number of retrieved items, both per information source and in total; the use of such limits is of practical benefit, to constrain memory requirements and processing time. Next, the minimal required status allows the user to concentrate on information that has been verified, or at least to remove from consideration facts that have been rejected, that is, marked as incorrect. Finally, TAKODO allows the user to specify a range of sentence numbers to limit research to specific passages within the document base. It is expected

that most users will ignore this last selectional criterion, but it was included as an aid during development and testing of TAKODO.

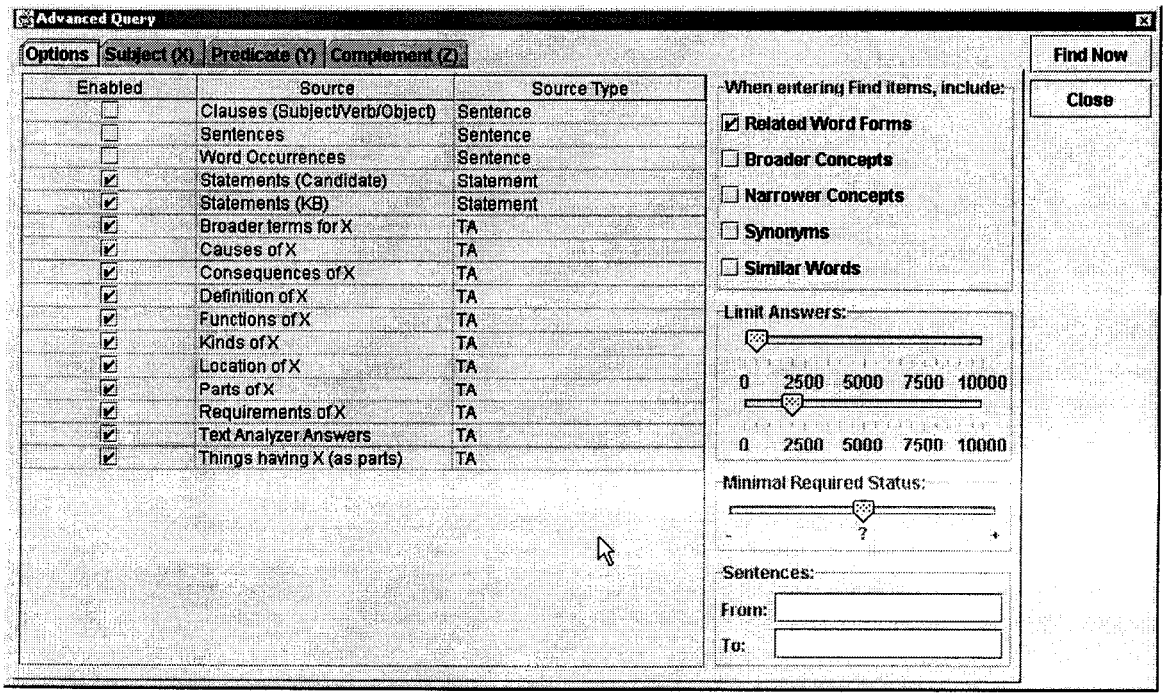



*Figure 28: Advanced Query Options in TAKODO*

| Query Expansion | Description |
|---|---|
| Related Word Forms | All inflected forms of a query term that are used in the document base, as identified by the automatic stemming algorithm. |
| Broader Concepts | All hypernyms of a query term (BTG thesaural relationship), as defined by the KB and the CKB. |
| Narrower Concepts | All hyponyms of a query term (NTG thesaural relationship), as defined by the KB and the CKB. |
| Synonyms | All synonyms of a query term (SYN thesaural relationship), as defined by the KB and the CKB. |
| Similar Words | All words in the document base that contain the query term as a substring (useful to find compounds, as well as other inflected forms, including those not automatically recognized). *In the future, may include related concepts obtained via spreading activation or other techniques.* |

*Table 15: Query Expansion Options in TAKODO*

Figure 29 presents the pane for one of the three parts of TAKODO's OAV query model (in this case, the Subject pane). The top half shows the term list and allows query expansion. All information sources are expected to accept, at minimum, a query

specification composed of a list of query terms; however, other selection criteria of a more specialized nature are allowed. It is not guaranteed that the latter will be interpreted by all information sources. Hence, for clarity, these have been grouped together at the bottom of the interface.

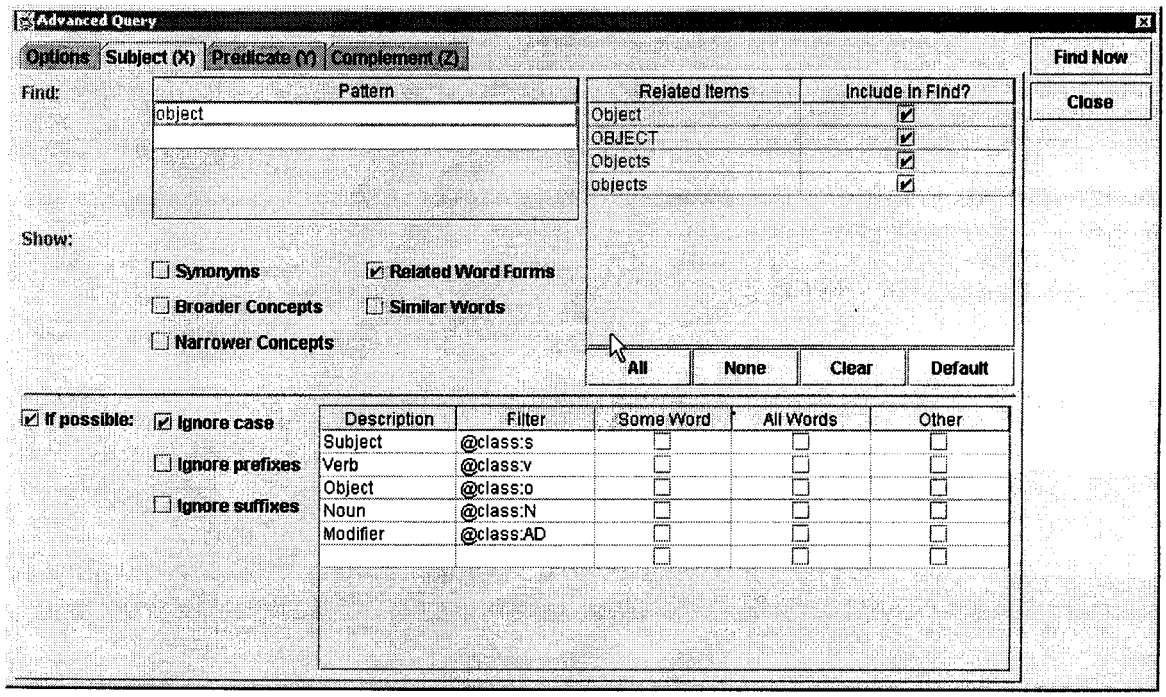



*Figure 29: Advanced Query in TAKODO*

### *4.7.5.1 Term List*

Ideally, the term list of the advanced query interface is a set of lexemes, although, by default, entries do not include a sense number. Nevertheless, the list can include any patterns, including words, compounds, or phrases. Here, the user may directly edit any term, add a term, or remove a term. Absence of any term is treated as a *blank*, that is, the user doesn't care about the value for the corresponding part of the OAV query model.

The interface allows manual intervention during query expansion. Associated with each term is another list of terms, or "related items." Selection of a query term and of query expansion options will result in the list of candidates for inclusion in the query. In this list of candidates, terms may be selectively included or removed, indicated by checkmarks.

Once again, both the term list and the "related items" grid displays may be sorted and filtered interactively, just as with the answers grid. For example, the user may show all hyponyms of a query term, say *vehicle*, and concentrate on a subset of these with an additional hierarchy filter, such as `-@isa:car`, to look at all known vehicles except cars (and their hyponyms).

#### *4.7.5.2 Additional Selectional Criteria*

Additional selectional criteria, which may not be supported by all information sources, include the option of ignoring case or performing term truncation. Even greater flexibility is given by allowing additional, custom criteria that are not specifically addressed by the query interface. A default collection of such criteria is presented in the bottom-most grid of the query interface, but the user may at any time add or remove items from this list. Each such item includes a description and a specification of the criteria, in the form of a filter.

Three checkboxes indicate how, if at all, the filter should be applied to a candidate relevant item. *Some Word* implies that, for any item that is a text passage, at least one word of the portion matching the query terms must satisfy the filter; *All Words* implies all words in the portion matching the query terms must satisfy the filter; and *Other* implies that the entire item, whatever it is (text passage, knowledge base statement, other), must satisfy the filter.

Thus, to find items about *object* where, in the case of a text passage, the word is used as the subject of a sentence, the user would type in the term in the term list, and enable the *Subject* filter (`@class:s`) for *Some Word*, or equivalently, *All Words* (both will have the same effect, since *object* is not a compound, thus any portion matching the query terms will consist of exactly one word).

## 4.8 Knowledge Management Features

### 4.8.1 Term Extraction

The *Words & Terms* panel (Figure 30) of the *Extras* pane is used to show words[56] (word-types) from the document base and terms from the knowledge base and CKB. Information for this panel is derived from a full-text inverted index; consequently, the list is quickly generated and allows unhindered interactive use.

By default, the display is sorted with highest frequency words first. A simple click of the mouse will toggle a word's state as being or not being a term, according to the knowledge bases[57] (including a third state of not making any such assertion about the word being a term). When there are several morphological variants, only one of these should be chosen as a term (akin to a headword in a dictionary).

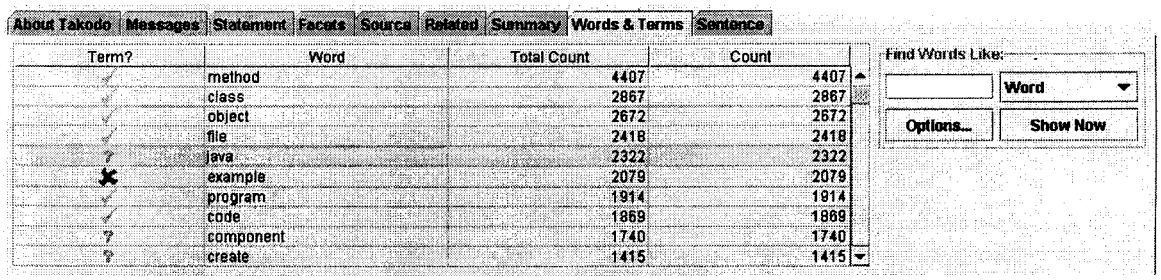



*Figure 30: Words & Terms Panel of TAKODO*

Several options (Figure 31) control what words or terms are shown, and how they are displayed. The user can specify minimum thresholds for occurrence statistics. Statistics can be computed using various levels of stemming or lemmatization, with word class restrictions; these results can then be displayed with different options for stemming or lemmatization.

---

[56] At present, the system does not include compounds in the *Words & Terms* panel. This will be addressed in a future upgrade, once the TA compound extraction and TAKODO compound extraction algorithms have been integrated.

[57] Information in the Candidates Knowledge Base (CKB) takes precedence over information in the knowledge base (kb). Asserting a word as a term will, if necessary, add a statement to the CKB, using the *term* predicate; retracting such a statement will remove the statement, or, if there is a prior statement in the kb asserting the term, will assert the rejection of this statement to the CKB.

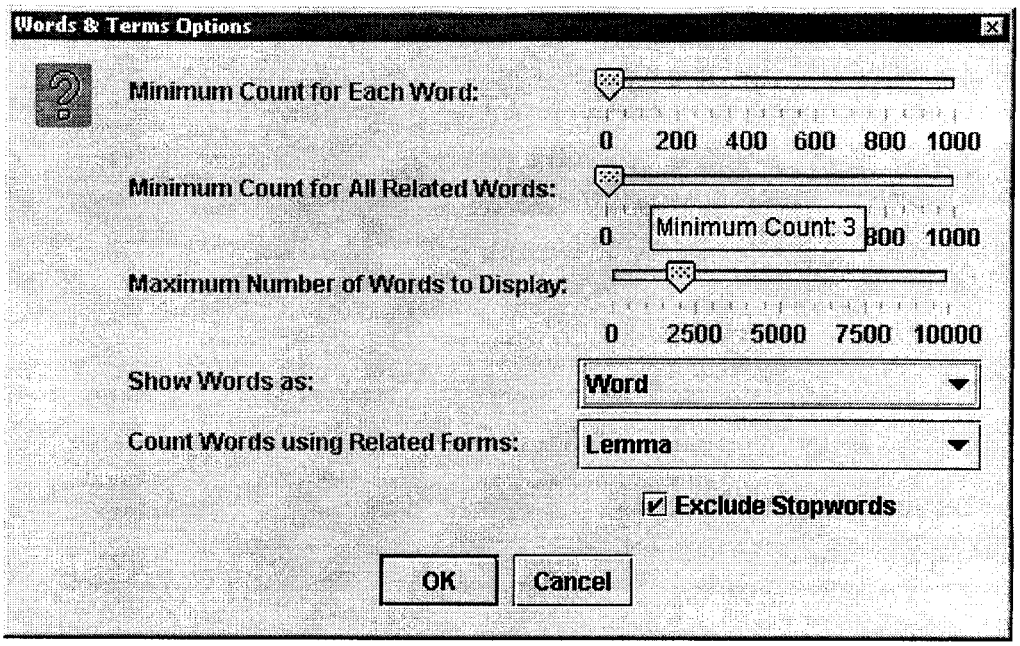



*Figure 31: Words & Terms Options in TAKODO*

For example, Figure 30 shows all content words, that is, all words not in the stoplist, lemmatized, and sorted in reverse order of frequency. Figure 32 shows all content words used as subjects, grouped by their lemma, and then sorted in reverse order of frequency. From these lists, generated from a document base about the Java programming language, one finds that good candidates for terms include *method*, *class*, and *program*, among others. In the second list, one immediately sees all morphological variants used in the document base, along with their frequency distributions. One may thus choose[58] among word variants to pick the most appropriate one for entry as a term.

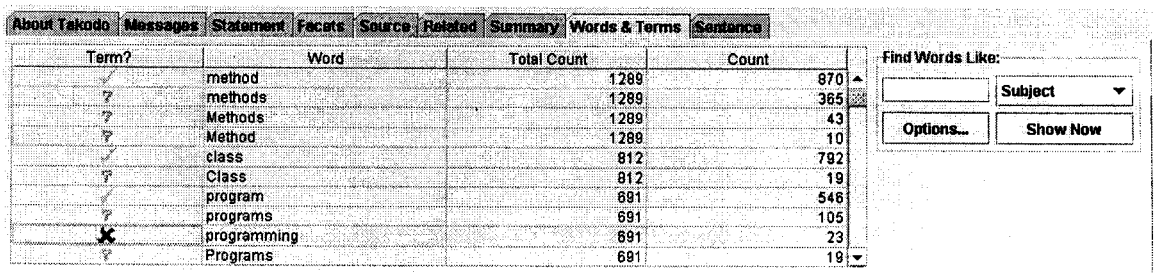



*Figure 32: Words Used as Subjects, Grouped by Lexeme, and Sorted by Frequency*

[58] This manual intervention can be used when automatic lemmatization fails to find the correct root form of a word.

Advanced filtering through the use of regular expressions allows morphological exploration to find terms. This use of regular expressions was advocated in [Heid et al., 1996]. Further analysis of a given word-type through observation of the word-tokens in context is achieved by querying for all word-occurrences, with the results presented using the concordance view of the answers grid.

### 4.8.2 Automated Extraction of a Taxonomy: An Experiment

Initially, it was the author's intent to use completely automated procedures to generate a taxonomy of concepts for the hyponymic and meronymic relationships, using the document base. There has been some previous work on the automated extraction of taxonomies from corpora [Grefenstette, 1994; Woods, 1997; Hearst, 1998; Sanderson & Croft, 1998]. Unfortunately, the initial experimentation did not provide the results the author hoped for. Subsequent research was deferred to future work; instead, the current version of TAKODO merely provides features to assist the manual creation of such taxonomies. The following provides some details of the experiment.

Processing began with the generation of a term list, in decreasing order of importance. This term list included any explicitly identified terms of the knowledge base and CKB, as well as words from the document base, using frequency of occurrence to measure importance, and assigning a minimal threshold.

Thereupon, the system iterated over the terms in the list. For each one, the TA was asked questions such as "What is the super of X?" or "What kinds of X are there?" The answers were recorded, generating a table of candidate hyponym/hypernym pairs (and other relations, during different trials of the experiment). Treating the pairs as co-occurrences, mutual information scores as well as z-scores were computed, to rank the results. It was hoped the highest-ranking result would be the most likely hypernym for the given hyponym. Unfortunately, this was not the case, and the results were of insufficient quality to merit their acceptance.

Manual inspection of the results showed particular difficulty in handling multiword terms, that is, compounds. This was traced to the answer extraction process within the version of TA used at that time. Prior experiments with the TA as an

information retrieval tool showed promise, because retrieval was measured based on the ability to find relevant sentences. Unfortunately, the algorithm to extract the answer from the retrieved sentence sometimes failed to adequately identify the correct fragment, especially when it consisted of more than one word. As discussed in the section on corpus linguistics, most terms are composed of multiple words.

A greater difficulty was due to morphological variations of the extracted answers, which adversely affected relevance ranking. The employment of a lemmatization process was intended to be used, to handle multiple inflected word forms. This proved considerably harder than expected, since the question answering process lost the tagging information from the initial preprocessing of source documents. Reconstructing this information would require tracing of the answers to their source sentence and searching for the extracted answer.

The author concluded at the time that the TA's answer extraction system would need improvement before being directly applicable to this task; indeed, there has been continued fine tuning of the patterns which may warrant another run of the experiment. If results are not satisfactory, the author believes a possible approach to address the problems is to ignore TA's answer extraction, and perform independent extraction using standard corpus-linguistic techniques such as co-occurrence statistics applied directly to the set of sentences retrieved by the TA.

### 4.8.3 Extraction of Taxonomies and Statements

Knowledge acquisition from corpora using TAKODO requires manual intervention. However, the system presents suggestions for entry into the knowledge base, and provides a user interface for accepting, rejecting, and manipulating suggestions in a very efficient manner.

Starting from the output of the *Words & Terms* panel, the user may identify important concepts. For this, the list of subjects or nouns provides the greatest insight. To initiate acquisition of knowledge about a given subject of interest, the chosen subject is pasted into the query interface. The user may ask for all information about this subject, or

refine the query to limit the results. In particular, common question types are easily accessed.

Figure 33 shows the result after asking for all broader terms (hypernyms) related to the concept *object*, as used in a corpus about the Java programming language. The first two answers were retrieved from the knowledge base[59], as evidenced by the source[60]. The following answers, in order of relevance, were generated by the question answering features of the TA. The first such answer states that "object isa array", which is incorrect. Selection of this answer to trace the source sentence reveals a situation where the inverse of the relationship holds (an array is an object), and that, it is possible, but not necessary, for a given object to be an array. Unfortunately, the TA did not expect the linguistic pattern used in the source sentence. The error is a product of the difficulties faced in natural language processing and natural language understanding, which hinder the complete automation of knowledge acquisition.

Since the answer was found incorrect, it may be immediately rejected with a single click of the mouse on the reject button (depicted as a thumbs-down icon on the toolbar). If possible, the acceptance or rejection of an answer updates its belief status. In the case of derived or otherwise non-editable answers, such as this example, the relevance judgement is logged by adding an equivalent statement to the CKB, recording the source, the user, and the desired belief status.

---

[59] The Java knowledge base is the work of the LAKE group and is used for instructional purposes. It was created prior to the completion of TAKODO, using the Text Analyzer, DocKMan's Knowledge Base Builder, and subsequently the Knowledge Organizer.

[60] The source "KB:WJKO" indicates that the answer derives from a Knowledge Base, using the Knowledge Organizer knowledge management system. During development, the acronym KO was used generically, to mean any knowledge management system, and the internal development name of the present Knowledge Organizer was WJKO.

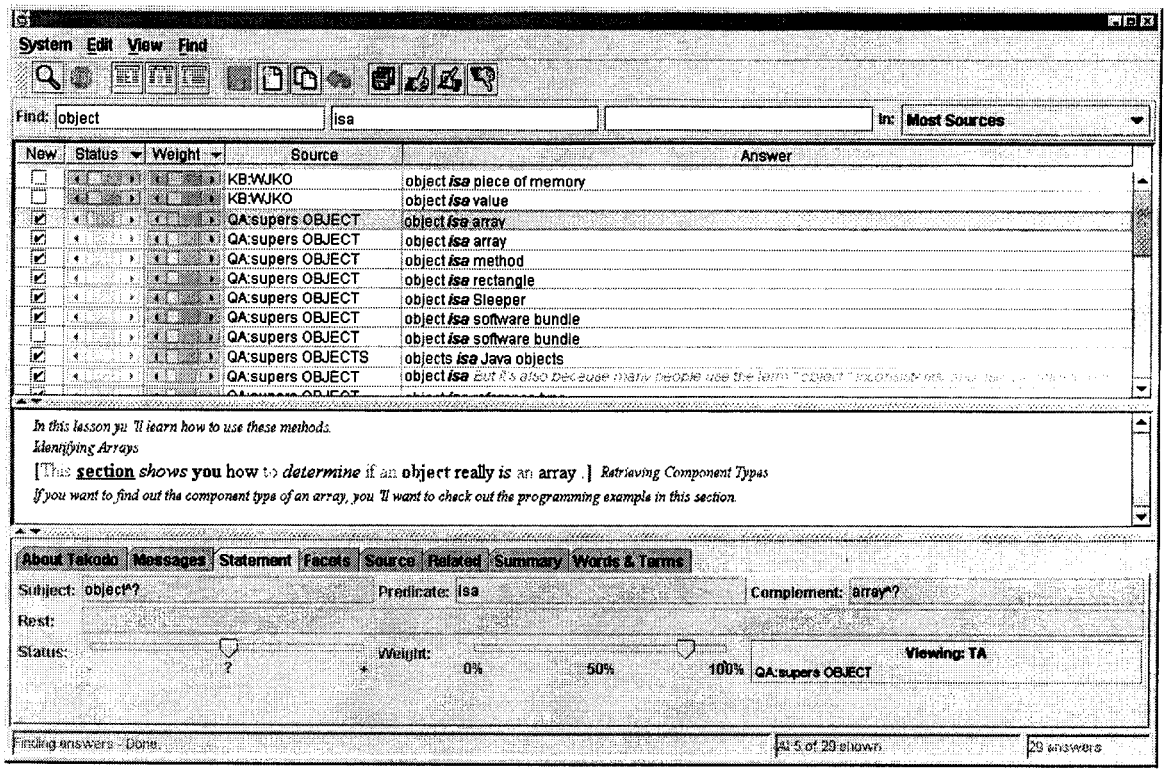



*Figure 33: Example of Knowledge Acquisition in TAKODO (Hypernyms of "object")*

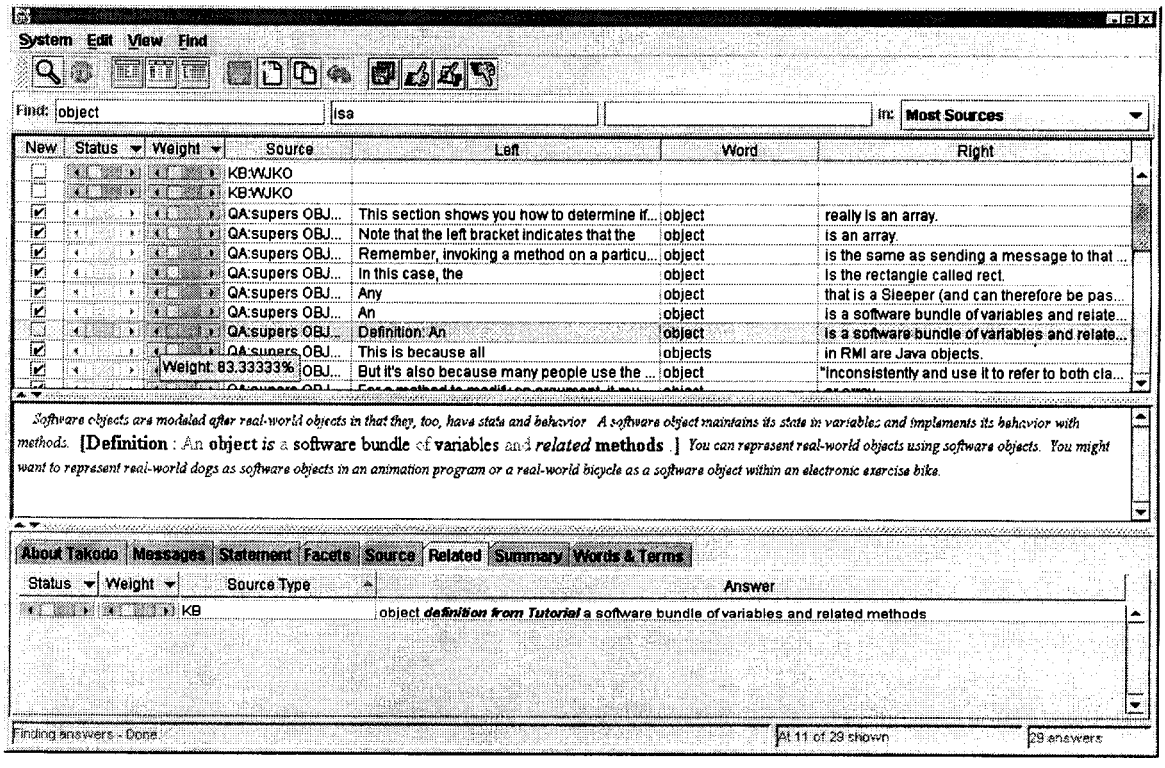



*Figure 34: Example (cont.) showing Concordance View & Related Answers*

#### *4.8.3.1 Knowing What is Already Known: The Related Pane*

When adding knowledge to a knowledge base, it is vitally important to know what is already known, to avoid duplication and prevent contradictions. TAKODO addresses this problem in two ways:

- The answers grid features a column called *New*, from which it is possible to identify the information is new with respect to the knowledge base. New answers are those without any related answers in the knowledge base (kb) or the CKB. These answers that have not been inspected for inclusion into the information base as stated knowledge are clearly distinguished from the rest. Filtering allows the user to restrict the display to new information during knowledge acquisition, or to eliminate from consideration these answers when the goal is one of knowledge verification.

- The *Related Pane* displays information from the knowledge bases (including CKB) that is related to the currently selected answer. At present, information is defined to be related if one of two conditions is met: (i) the information has a similar subject, and, if defined, a similar predicate and complement, using a case insensitive match ignoring sense numbers; or (ii) the auditing information states that the information was derived from the selected answer. This auditing information includes the source sentence and any unique identifier[61] (ID) that may be associated with the answer. In the future, one may extend the definition of relatedness and provide user customization for the kinds of related information that is displayed.

Continuing the process of knowledge acquisition from the previous example, let us inspect the 9th answer, "object isa software bundle," which had been inspected during a prior knowledge acquisition session. Hence, the *New* column is unchecked. In Figure 34, the user has dynamically switched to a concordance view[62]. The *Related Pane* shows

[61] All knowledge base elements in the Knowledge Organizer have unique object identifiers (OID). Many TAKODO knowledge base elements have similar unique OIDs.

[62] The concordance view is appropriate for answers that are derived from the document base. Answers without a link to source passages of text will be displayed as empty cells for the concordance columns.

a statement that was entered into the KB using the source sentence. As it turns out, one finds a definition of an object. There is no statement relating *software bundles* as the hypernym of *object*, as the authors of the knowledge base preferred, instead, the genus *piece of memory*.

#### *4.8.3.2 Navigation via Browsing of Related Answers*

Answers in the *Related Pane* may be sent to the answers grid, replacing its contents and updating the *Related Pane* with information related to the new content. This process may be viewed as navigation, much as in a hypermedia system.

#### *4.8.3.3 Summary Statistics*

The number of answers can be quite large. In order to summarize the information, TAKODO offers the *Summary Pane*. This pane can be used to calculate frequency of occurrence statistics for various columns of the answers grid. High frequency may mean unnecessary duplication, such as the same statement entered twice into the knowledge base. It may also mean higher relevance, such as several sentences from the document base repeating the same basic assertion.

Figure 35 shows the distinct subject-predicate-complement statement triples used by the answers of the previous example, sorted by frequency. One can quickly see that three statements appear twice[63].

[63] Two statements appear once each in the knowledge base and as an answer from the TA. The last repeated statement appears from two source sentences that were found relevant by the TA.

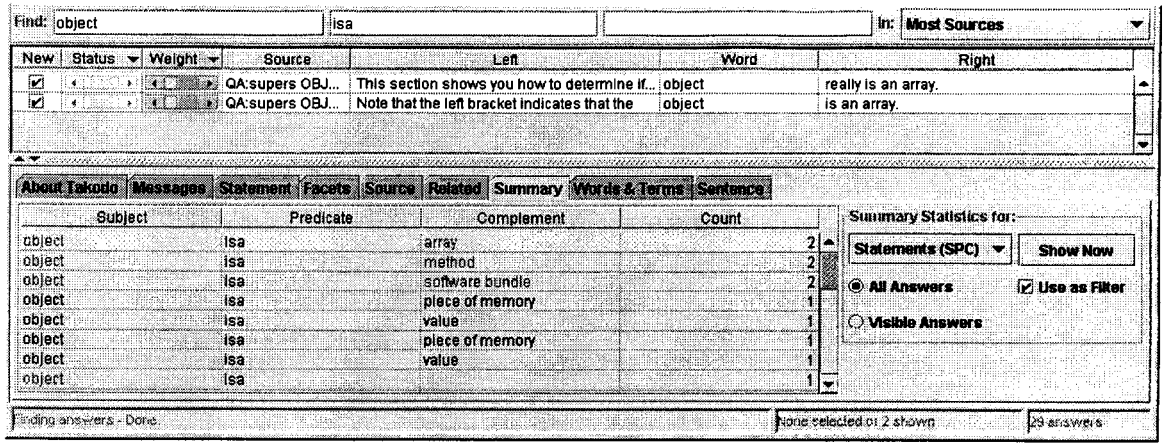



*Figure 35: Summary Statistics for Answers in TAKODO*

The figure also shows yet another filtering option: the answers grid has been restricted to statements having the selected subject, predicate, and complement. The display is limited to only those statements asserting that an object is an array.

The combination of summary statistics with the filtering option is very effective for browsing answers in distinct groups. A simple query can be specified to find all information about a given subject, and the *Summary Pane* set to show all distinct predicates in order of frequency. Such a display shows the highest correlated predicates for the subject, in effect suggesting important statements for entry into the knowledge base. Simultaneously, the user may browse suggestions of the appropriate complement for use in any such statements.

#### *4.8.3.4 Editing*

Some answers, such as those derived from the document base or inferred from a knowledge base, exist merely to provide information and may not be edited. Those which can be edited, such as stated kb statements, may have restrictions regarding which parts are editable or which users have permission to make edits. The exact nature of the restrictions depends on the information source. For example, KO has a security and authentication model that is distinct from its predecessor, DKBB, and different systems may enforce different integrity constraints. Through an abstract API, TAKODO handles all of these cases. The editability of an answer or of its parts is reflected in the user interface through the toolbar, menu, and editing panes of the *Extras* pane.

Edits to a single answer may be performed directly in-place, by entering new values into the answers grid. Equivalently, edits may be performed in the *Statement* pane (Figure 36), *Facets* pane (Figure 37), and *Source* pane (Figure 38) of the *Extras* pane. Any changes are immediately reflected throughout the user interface.

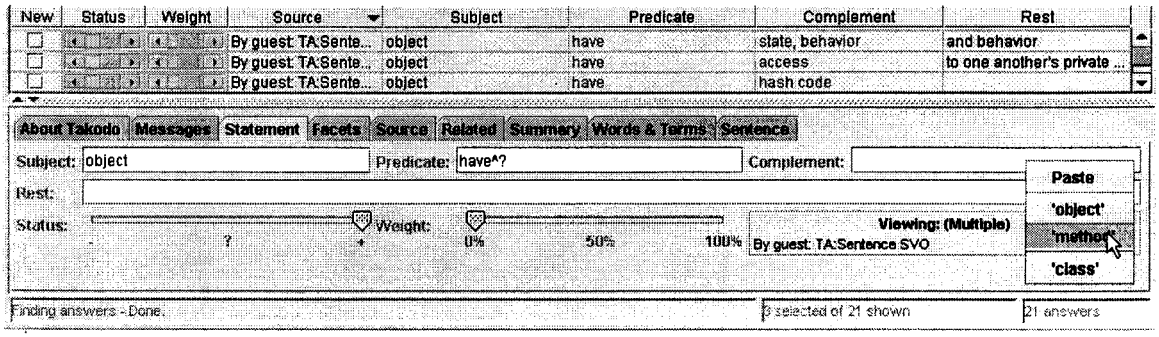



*Figure 36: Statement Pane of TAKODO (Bulk Editing of Three Answers)*

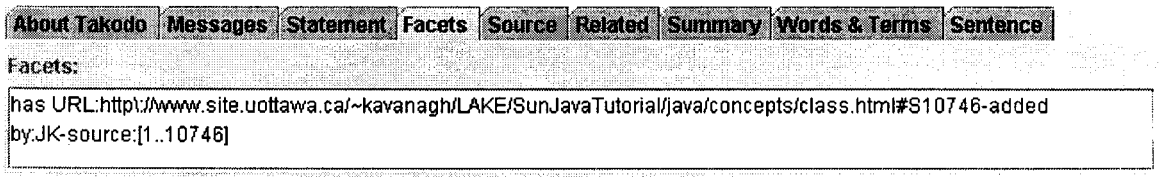



*Figure 37: Facets Pane of TAKODO*

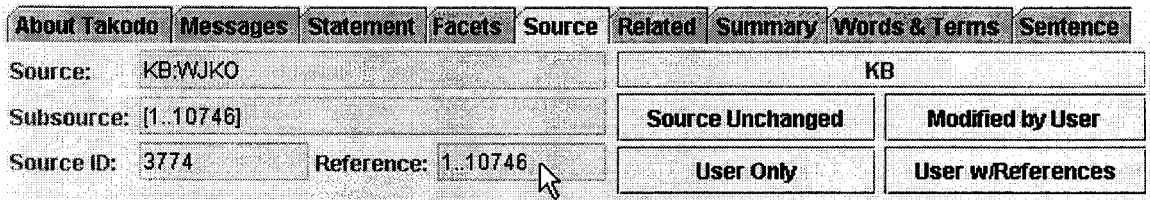



*Figure 38: Source Pane of TAKODO*

Bulk editing operations are supported through the simultaneous updating of all selected answers. For these, the user may modify only those parts that are uniformly editable in all selected answers. Any failures encountered during the editing transactions are signaled to the user. Situations leading to such a failure include violations of integrity constraints, for example, by attempting to introduce cycles into a taxonomy that specifically forbids them.

Numerous productivity enhancements allow the user to copy and paste information between the various user interface elements of TAKODO. Figure 36 showed

a popup list of suggested values for entry into the complement of a statement. Figure 39 shows a user editing a different statement, after having updated the subject and complement. Now, a passage from the *Sentence* pane is being sent to the statement's rest. The transfer will update the audit trail to record a reference to the source sentence.

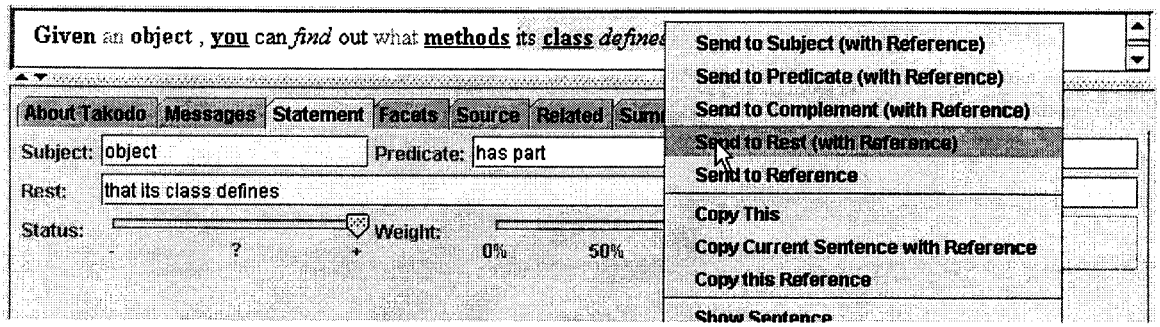



*Figure 39: Editing of a Statement in TAKODO*

In fact, every edit, whether it is to create a new statement in the knowledge base or to update an existing answer, will record the user effecting the change, for audit tracing purposes.

#### *4.8.3.5 Finalizing a Taxonomy*

As discussed, TAKODO allows knowledge bases to contain inconsistent information. This support is continued in the CKB knowledge base. In contrast, systems such as KO enforce specific integrity constraints on the hyponymic and meronymic taxonomies. For example, KO disallows cycles. In addition, KO's predecessor, DKBB, requires all subjects to be placed under some parent subject.

To assist the user in generating a taxonomy that is suitable for use in these other systems, TAKODO includes an automatic repair algorithm. This algorithm proceeds in three phases. The first identifies cycles and breaks them arbitrarily but with preference for weaker links, according the their belief status and weight. The second phase places any unparented subjects under a specified top parent (*Unparented* for KO or *kbTop* for

DKBB). The third stage removes any links that are derivable from the transitive[64] nature of hierarchical relationships.

The automated algorithm does not guarantee a conceptually correct taxonomy. For this, manual intervention is required. Presently, TAKODO's interface is not ideally suited to interactive manipulation of the hierarchies such as the conceptual taxonomy. The KO, particularly with its graphical visualization of the knowledge base as a semantic network, is a considerably more appropriate tool. The author hopes to further integrate elements of the KO with those of TAKODO to provide the best of both worlds.

[64] For example, if *a* is a hypernym of *b*, and *b* is a hypernym of *c*, then by transitivity, *a* is a hypernym of *c*. Recording the last fact in a knowledge base as *a priori* knowledge is redundant, since the inference engine will deduce the required *inferred* knowledge.

# Chapter 5: Implementation

The design and implementation of the TAKODO core, which excludes external systems such as TA and KO, was solely the responsibility of the author of this thesis. The system was targeted towards the Java 2 Platform, currently[65] using the Java Development Kit (JDK) version 1.3 Release Candidate 1. The TAKODO core consists of approximately 340 classes, organized into 45 distinct Java *packages*[66] and totaling over 1.54 megabytes of source code. Given the complexity of the code, a complete description of implementation details is beyond the scope of this thesis.

Beyond the listed external systems, TAKODO currently interacts directly with two relational database management systems (RDBMS): Oracle Corporation's Oracle version 8, communicating through Oracle's Thin JDBC driver; and, for legacy support of DocKMan's Knowledge Base Builder, Sybase Corporation's Sybase, communicating through Symantec Corporation's dbAnywhere JDBC driver.

Throughout the following discussions, the reader is encouraged to refer to the data-flow diagram of the TAKODO system, as depicted in Figure 40. Parts of this diagram are described summarily in Table 16.

[65] Development began with JDK version 1.2 beta 2 and continued throughout various beta and final releases during the evolution of the Java 2 Platform.

[66] Packages in Java form a nested organizational structure with implications for namespace resolution and encapsulation, and are used for software modularization. A package is the smallest unit used to implement a complete software specification.

| System | Module | Description |
|---|---|---|
| TAKODO | TAKO | Combines into a unified interface the linguistic, inferencing, message delivery, querying, and filtering functions of TAKODO, delegating to other subsystems as necessary. |
| | Takodo TA | Provides text analysis functions, including:<br>• Linguistic services, such as lemmatization.<br>• Statistics, such as word or co-occurrence frequencies.<br>• Question answering, through delegation to the TA's Q&A Server. |
| | Takodo KO | Provides services to access knowledge bases, defining a generic, abstract interface to knowledge management systems. Concrete implementations provide communication with systems such as the Knowledge Organizer (KO), among others. |
| | Candidates | Provides a scaleable solution for the storage and retrieval of various kinds of *statements* (all of which share a common logical model as OAV triplets):<br>• Manages transient and persistent answers from question answering using the TA.<br>• Implements a knowledge management system containing the persistent statements constituting the Candidates Knowledge Base (CKB). |
| | Takodo Oracle | Provides an abstract interface for performing information retrieval, and defines concrete implementations for retrieval from knowledge bases (CKB, KO, etc.) and the document base (using sentences, clauses, and TA's question answering services). |
| Text Analyzer (TA) | TA Application | Independent Visual Basic application, which provides text analysis functions. Used by TAKODO core. |
| | Q&A Server | Part of TA Application. Provides question answering features as a server using a socket-based communications protocol. |
| | Preprocessor | Preprocesses source documents (for example, morphosyntactic analysis) and stores results into a database for subsequent use by the TA Application. |
| Knowledge Organizer (KO) | | Independent frame-based knowledge management system. Used by TAKODO core |

*Table 16: Main Systems and Modules of TAKODO*

## 5.1 Preprocessing

Preprocessing begins with the use of the Text Analyzer's preprocessing system, which has already been described. The TA performs tokenization of source documents (as text or HTML), including word and sentence boundary determination. Morphosyntactic analysis, part-of speech tagging, and limited lemmatization (restricted to the identification of infinitive forms for many conjugated verbs) are performed by ENGCG. Information is re-represented as one sentence per database row, using TA notation for the tagging

information, and imported into the Oracle database. Some summary statistics such as sentence and document length are also generated.

### 5.1.1 Initialization

On startup of TAKODO, the database schema is inspected to verify whether the database has been initialized for TAKODO's use. If not, any incomplete initialization or preprocessing is resumed.

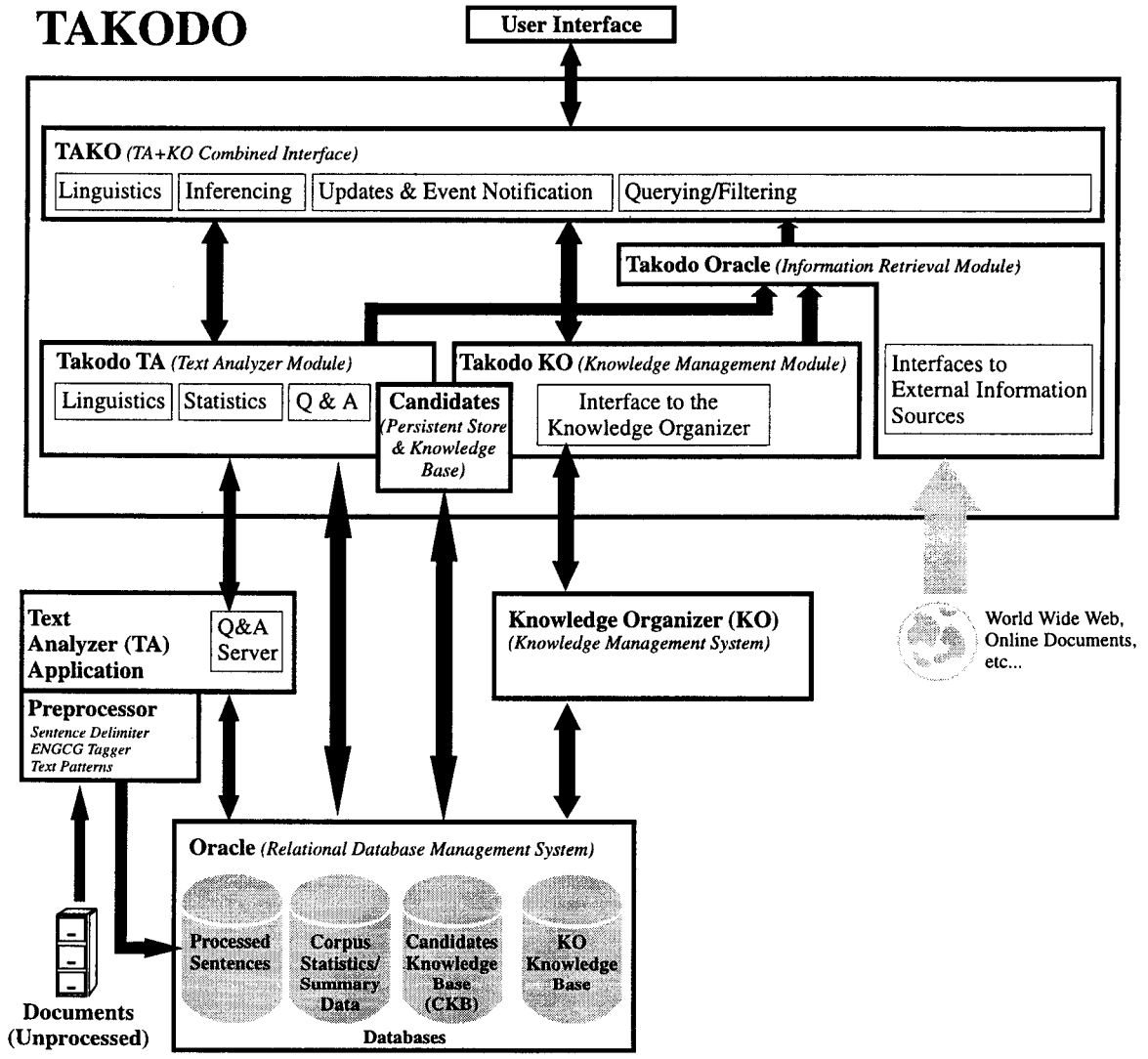



*Figure 40: Data Flow Diagram of TAKODO Implementation*

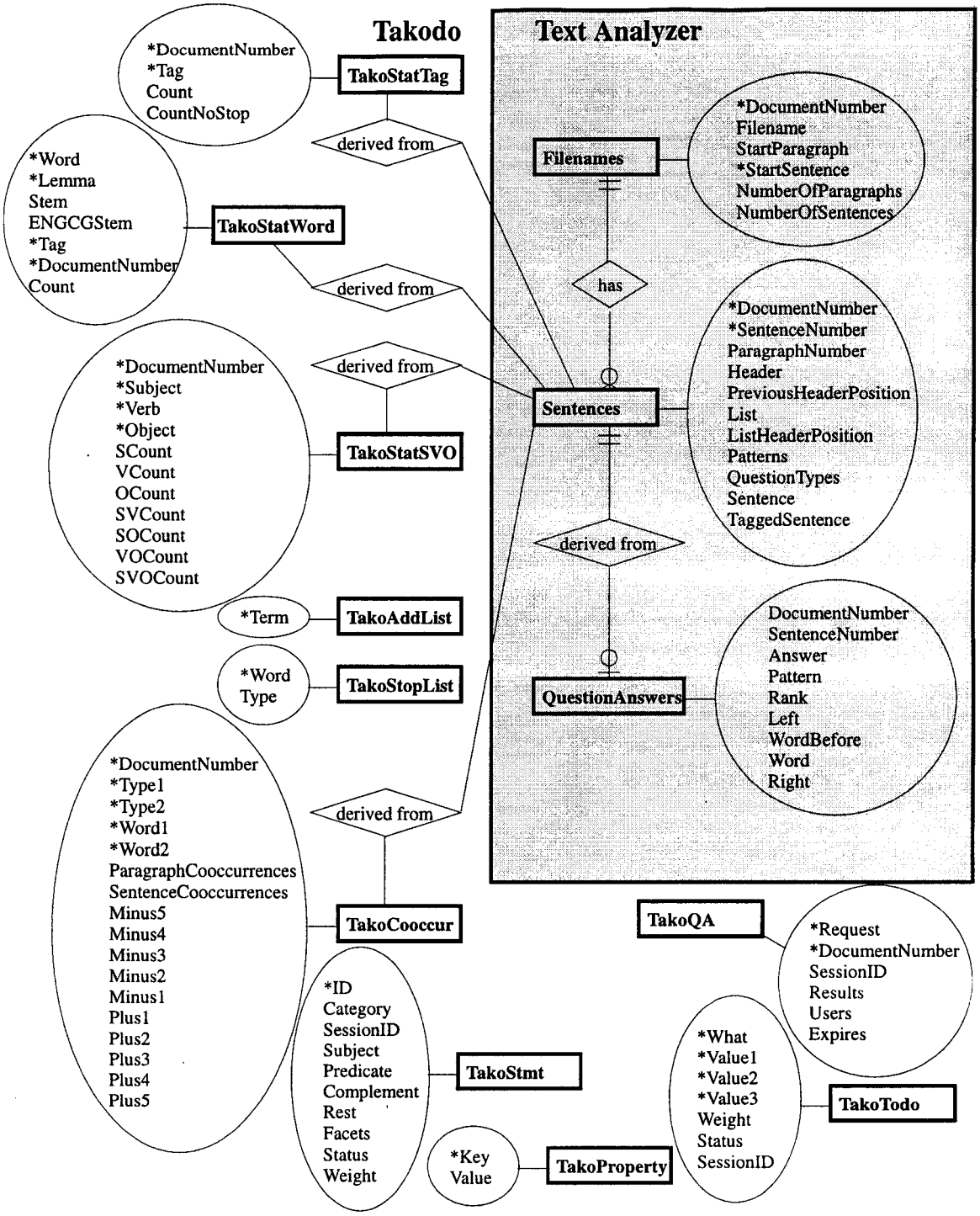



*Figure 41: Entity-Relationship Diagram of TAKODO Tables in the Database Schema*

Initialization includes updating the database schema to include the necessary tables for use by TAKODO. These tables, and some of the more relevant ones from the TA, are presented in Figure 41. For brevity, this thesis omits a full description of database indexes and several other performance-tuning concerns.

The *TakoProperty* and *TakoTodo* tables record global state information regarding the TAKODO information base. Of these two, the second maintains the progress of knowledge acquisition tasks, particularly during experimentation on the automated extraction of a conceptual taxonomy.

*TakoQA* records state information regarding the answer cache, which is initially empty. This cache is a performance improvement for frequently asked questions (FAQs), and stores frequently needed TA answers in the *TakoStmt* table. The table clusters information for efficient retrieval, reducing latency overhead from random seeks, thus maximizing throughput.

*TakoAddList* and *TakoStopList* are used for stopwording during full-text indexing (described later) and other such processing. A morphologically driven algorithm[67] is used for initial identification of stopwords. Any word that does not contain at least one alphabetic character or that exceeds a maximum length threshold is, by default, marked as a stopword. *TakoStopList* contains an additional exclusion list of stopwords, and *TakoAddList* records exceptions to the automatic stopwording of the morphological algorithm. If these two tables have not been initialized, then the former is automatically filled with a default stoplist (Table 18 of the Appendix), and the latter is left empty.

[67] This algorithm was found appropriate for use with the research corpora regarding the Java language. This corpora included embedded chunks of source code for illustrative purposes, including many occurrences of unimportant numbers, unusual symbols, and very long tokens used as program identifiers.

### 5.1.2 Full-Text Indexing of the Document Base

In order to address computational efficiency during retrieval, both for interactive ad-hoc queries by the user, and most importantly for the derivation of summary frequency of occurrence statistics during other stages of TAKODO's processing, the author decided to employ full-text indexing of the document base. Note that the TA, as used by TAKODO, does not currently make use of full-text indexing, although this is being added to address performance issues.

Originally, the author intended to use Oracle's ConText Cartridge as the indexing engine. The LAKE group's initial experiments with the TA, as already discussed, had shown some disadvantages in terms of administrative overhead and retrieval performance, particularly for high-frequency words. As yet, the group is unsure whether the poor performance was the result of incorrect tuning of the engine's and database's parameters, or whether it is the database schema that is the problem. The storage of documents as individual sentences in distinct rows of a database is certainly unusual.

Given the performance goals of TAKODO, and the need for term lists and additional summary statistics for relevance ranking, the author felt it was warranted to re-evaluate the decision to disregard ConText. Unfortunately, further inspection revealed ConText to be inappropriate for TAKODO's needs. The ConText engine is a complete information retrieval system, with its own query expansion and semantic analysis technology. It is also an insufficiently open system, which does not make its lower level functions available for external modification or reuse. Although it is possible to designate a thesaurus and stoplist for use by the system, it is not possible to directly intercede in the indexing, query expansion, or relevance ranking process. Portions such as the stemming algorithm remain inaccessible outside of information retrieval by ConText. Since TAKODO must make such linguistic services available to a wide range of information sources, such as the TA (which, for example, does not perform word stemming), the

author decided[68] to provide a custom-designed, minimal implementation of the desired functionality.

*TakoStatWord* contains the full-text index of all words in the document base, except those specified by the stoplist. Index entries are grouped by their lemmas, allowing efficient access to morphological variations. Granularity is at the document level, and statistics on syntactic and part-of-speech tags are maintained, for use in computing such values as tf·idf weights. Efficient storage of this table is the responsibility of the Oracle RDBMS. The database implementation makes use of B-trees as the underlying data structure, so retrieval time should be logarithmic in the number of words.

During the indexing process, summary statistics counting the total number of distinct word classes are maintained and the *TakoStatTag* table updated accordingly. This information is used during the computation of relevance and similarity scores, for the normalization of the values.

At any time, the inverted index may be updated incrementally with additions and removal of documents to and from the document base.

### 5.1.3 Statistical Analysis

Co-occurrence statistics may be computed from the document base. *TakoStatSVO* contains subject-verb-object co-occurrence statistics listing distinct triplets and their frequency of occurrence. *TakoCooccur* is used for collocational analysis, similar to that used in Xtract [Smadja, 1993]. This collocational analysis may be based on bigrams of inflected word forms or on any lemmatized form thereof. The design of *TakoCooccur* further supports colligational analysis by allowing the specification of word class bigrams.

### 5.1.4 Frequently Asked Questions (FAQs)

*TakoStmt* is an Oracle *partitioned* database table. In other words, it is a logical table containing conceptually and structurally similar records but with distinct purposes

[68] Few information retrieval systems are able to directly support documents stored in a database. Investigations of publicly available code libraries did not find any that completely satisfied TAKODO's requirements, the cost and effort of acquiring, and possibly modifying, a suitable system from a third party source were evaluated as excessive.

and different performance characteristics. Information stored in the table is separately clustered by Oracle based on criteria specified in the database schema.

One use of *TakoStmt* is to cache frequently asked questions and their corresponding answers from the TA, clustering the resulting answers as stored in secondary storage for increased throughput and reduced latency. Whenever a TA question is requested, TAKODO first inspects the cache for the presence of the answers. If the answers are found in the cache, then their lease times are renewed. Otherwise, the request is forwarded to the Text Analyzer server, and the results appended to the cache. The cache is maintained consistent across multi-user transactions, and uses a hybrid replacement policy with renewable timed leases and reference counted garbage collection to prevent answers from being removed prematurely while in use. The lease times approximate a Least Recently Used (LRU) replacement policy. Also, a limit on the maximum cache size is imposed.

The cache allows very important answers to be recorded permanently, thus avoiding the subsequent need to request the same information from the TA during interactive use of TAKODO. This permanent caching of TA answers may be requested manually subsequent to query processing. Alternatively, in the experiment regarding the automated construction of a hyponymic taxonomy, the *TakoStmt* table was used to store answers of several question types related to important words in the document base, as identified by frequency of occurrence statistics.

### 5.1.5 Indexing of the Knowledge Base

The *TakoStmt* table is also used to store the Candidates Knowledge Base (CKB). For efficient retrieval, the table is indexed by subject, predicate, and complement. The indexes are managed by the Oracle RDBMS.

## 5.2 Other Processing and User Interface Transformations

### 5.2.1 Parsing and Processing of Sentences

Rather than use a markup language, TAKODO uses an object-oriented interface to model sentences. Conversion of sentences from the tagged notation as used by TA into

the objects representing the sentences requires a parsing step. The parser was hand-coded and contains special handling for ambiguity[69] resolution and the correction of some consistent tagging errors.

The TAKODO model provides several linguistic facilities including more sophisticated lemmatization. Several choices for lemmatization are offered: stemming of the original word via the Porter stemming algorithm[70]; stemming of ENGCG lemmas; and a combination of case normalization, stemming, and term truncation (to a maximum length of 5 characters). The last option is sometimes used for hashing purposes or when there is a need to cluster similar words.

TAKODO does not perform any automated word sense disambiguation. The limited benefits for retrieval have been discussed previously in Chapter 2. Note that the knowledge base does include optional sense numbers, and TAKODO was designed to accommodate future integration of a word sense disambiguation algorithm.

### 5.2.2 Identifying Clauses

Note that, for efficiency reasons, deep natural language parsing of sentences was avoided; instead, TAKODO uses only surface morphosyntactic analysis. This analysis, using the TA and ENGCG, tags, among other parts of speech, the syntactic and, to a limited extent, semantic subjects, verbs, and objects; unfortunately, the output of this analysis does not explicitly identify how these are related into clauses. Instead, TAKODO employs a heuristic based on the observation that 90% of clauses in Modern English containing a syntactic subject, verb, and object will contain these in exactly that order [Crystal, 1997].

Therefore, the algorithm to identify clauses begins by finding a word tagged as a verb, and searching backwards for the first word tagged as a subject, and forwards for the first word tagged as a direct object. In both cases, if none is found in the initial direction,

---

[69] Due to some minor problems with the tagging notation and errors introduced by the TA whilst generating its output, the tagged sentences contain ambiguities in the interpretation of tag delimiters, especially when the textual content of a document contains the same characters as these delimiters.

[70] A freeware implementation of the Porter stemming algorithm [Lazarinis, 1997] was used.

the other direction is then searched. If no word tagged as direct object is found, the algorithm searches for a word tagged as indirect object.

### 5.2.3 Representational Issues

KO's predecessor, DKBB, uses different naming conventions for representing information than does the current system. For legacy support, the interface between TAKODO and DKBB performs some limited dynamic conversions by renaming certain predicates and facets to match the TAKODO conventions, for example, to change `part` or `parts` into `has part`. Some parts of a statement, such as belief values, have explicit support in TAKODO but are stored as facets in these knowledge bases. Currently, the exact nature of the transformations is fixed and hard-coded, and a more flexible approach is desirable.

## *5.3 Architectural Design and Object-Oriented Decomposition*

The appendix contains some abridged UML class diagrams of the TAKODO system. Several main classes are presented in Figure 47.

### 5.3.1 Implementation Platform

To meet the reusability goals of TAKODO, the author wanted to retain as much platform-independence as possible. To this end, the author chose the Java 2 Platform as the implementation target, allows deployment to various environments, from Unix to Macintosh to Windows, and several more. The Java White Paper marketing documentation describes Java as a simple, object-oriented, distributed, dynamic, multithreaded, high-performance, robust, secure, architecture neutral and portable system. Whatever inaccuracies may lie with respect to the use of these buzzwords, the degree of vendor and platform support for Java are certainly very significant.

In order to service secondary storage needs, TAKODO uses Oracle Corporation's Oracle version 8 RDBMS. The choice stems mainly from the reliance on the TA. Use of a common RDBMS provided greater ease of accessibility to the information base. Vendor support and quality of the product (in terms of efficiency, stability, and robustness)

provided further motivation for the choice. TAKODO makes use of Oracle-specific performance-enhancing features, such as *partitioned* tables.

For legacy support of DKBB knowledge bases, TAKODO directly manipulates the databases managed by Sybase Corporation's Sybase RDBMS.

### 5.3.2 Componentware for Reusability and Maintainability

TAKODO has been created to make effective use of good object-oriented design principles in order to achieve an extensible, maintainable architecture suitable for use as a framework allowing future development. However, object-oriented programming (OOP) languages have not achieved all of the benefits that proponents claimed, and instead developers are looking towards componentware to address software reusability issues [Microsoft Corporation, 1997]. To quote from the JavaBean FAQ[71]: "Developers are turning to creating components rather than monolithic applications to free themselves from slow, expensive application development, and to build up a portable, reusable code base." A *component* is a "reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application."[72] Unlike objects, components are supported at the system software level, rather than programming language/development environment level [Cottman, 1998].

*JavaBeans* is Java's specification for a component model, providing the component-aware object-enabling system software as well as the design guidelines required for component construction. A Java *bean* (note the lowercase) is a reusable software component conforming to the JavaBeans specification. Also according to the JavaBeans FAQ, beans are typically characterized by an ability for introspection to allow analysis of the bean's external behavior and API, customization support, provisions for event notification, the existence of mutable or immutable properties, and some form of persistence.

---

[71] Available: http://www.javasoft.com/beans/faq/faq.general.html

[72] what?is. [Online] Available: http://www.whatis.com/

Component development in Java is greatly simplified as compared to other development environments. Depending on the required functionality, the difference between the creation of an object versus the creation of a bean can be as minor as conformance to JavaBeans naming conventions. This is especially true for very simple components.

Within TAKODO, both the information itself and the information providers are modeled using the JavaBeans componentware model. Thus, an answer from the answers table is really a Java bean. A query specification is a Java bean. The Text Analyzer information source is a Java bean. Since they are all beans, they may specify their own computational behavior, and even include their own user interfaces for customization.

The use of Java beans for answers is especially important. Rather than explicitly store all information about the answer, as would be needed by a simpler data structure or segment of marked-up text, the answer bean may compute information whenever it is needed. It is this behavior that allows the dynamic changes to visualization options, going from a concordance view to a statement view with extraction of a subject, predicate, and complement.

### 5.3.3 Persistent Storage Mechanisms

TAKODO, including its related systems such as KO, manages the persistent storage of various information items. Chiefly, among these, are the statements of the Candidates Knowledge Base (CKB). The author decided to use an RDBMS because (i) the relational model, with its flexible handling of tables of data, accommodates much of the required linguistic and statistical processing; (ii) such systems are designed for scalability and multi-user access; and (iii) it is a mature technology, with wide vendor and application support, and standardization of language bindings, such as JDBC, and of query syntax, via the Structured Query Language (SQL).

Object-Oriented Database Management Systems (OODBMS) and the hybrids, Object-Relational Database Management Systems (ORDBMS), are just in the process of maturation. For example, Oracle is an ORDBMS but many of its object-oriented extensions are proprietary. Queries still employ SQL, with special additions, rather than

the Object Query Language (OQL) being promulgated by the Object Data Management Group[73] (ODMG). Sun Microsystems has approved[74] the development of software to address the need for "a standard way to store Java objects persistently in transactional data stores [... and] a standard way to treat relational database data as Java objects." The ODMG has recently submitted its recommendations to this development.

Given the discussion of future development in the area of persistent storage, let us at least mention the existence of a project, SQL/MM Part2: Full Text, to add explicit support for structured text to RDBMS technology [Davis, 1996]. The specification addresses needs for retrieval, to find textual patterns, and the markup of fragments.

TAKODO's persistent storage mechanisms must compensate for several deficiencies of relational databases in the context of knowledge-based systems [Wagner, 1999], some of which have been addressed by database vendors, such as Oracle. Wagner recommends several extensions, some of which are listed below:

- *Unique and immutable object identifiers (OIDs)*: Identification of rows in an RDBMS employs unique primary keys. When no natural keys exist for an entity, artificial keys must be generated. Also, primary keys are mutable, which may be problematic. TAKODO generates its own OIDs to manage data.

- *Complex-valued attributes*: The first normal form requires complex attributes to be split into their constituents rather than treated as a whole. TAKODO assembles and disassembles complex-valued attributes during the transport from and to the database.

- *Attribute functions*: Information that is computed is poorly handled, and must be explicitly stored in the database if it is subject to certain kinds of processing. For example, the inverted index stores rather than computes lemmatized and stemmed

[73] Available: http://www.odmg.org/

[74] JSR-000012 Java Data Objects (JDO) Specification.
Available: http://java.sun.com/aboutJava/communityprocess/jsr/jsr_012_dataobj.html

word forms, so that indexing will cluster morphological variations. This increases space consumption.

- *Subclass hierarchy and inheritance*: Hierarchical data is not supported by many RDBMS systems; Oracle provides limited, nonstandard extensions. TAKODO must provide its own inheritance and inferencing.

- *User-defined types and large objects*: Binary Large Objects (BLOBs) are an inadequate solution for abstract data types (ADTs).

- *Reaction rules*: Database triggers offer limited functionality. For example, event notification is problematic: in a distributed processing environment with multiple, concurrent users, it is hard to monitor changes to data, so as to maintain the presentation of information up-to-date in the user interface.

- *Security*: Security is at the schema level. An RDBMS does not support the fine-grained security needs of knowledge-based systems, where restrictions depend on the data's content. For example, one may wish to limit a user's updates to a subset of the conceptual taxonomy [Prpic, 1997a].

- *Uncertainty and reliability*: There is no explicit support. TAKODO adds its own extensions to record belief values.

TAKODO employs conventional object-relational mapping techniques to deal with the object-relational impedance mismatch [Ambler, 1997]. TAKODO defines a domain-specific query mechanism that hides many underlying retrieval implementation details and minimizes reliance on the Structured Query Language (SQL) used by the DBMS system. Caching techniques attempt to maintain frequently used objects in memory, to avoid the performance penalties associated with access to secondary storage. Such performance penalties are otherwise frequent when retrieval is essentially navigational, as the relational model handles navigational access poorly. Some details of the Knowledge Organizer's caching system have already been covered. Additional caching within TAKODO relies on Java's garbage collection mechanism, using *weak* references that do not impact which objects are eligible for reclamation. Of course, the

storage mechanism must make use of appropriate data structures and indexing strategies to ensure high performance. These employ the standard Java Collections classes, including hash-based indexes. In addition, the storage mechanism was designed to be thread-safe, given the use of concurrency in TAKODO. Due to the complexity inherent in all of these issues and the space constraints of this thesis, a description of the storage mechanism in greater detail has been avoided.

### 5.3.4 User Interface using Model-View-Controller Architecture

The user interface is separated from the core knowledge management systems. Communication between the two is assured by using a Model-View-Controller design pattern. Coupling between the systems is limited by using abstract interfaces; information is kept synchronized and consistent through an event notification system that broadcasts changes as necessary. This event notification system is implemented using a combination of the standard Java Swing event model, the standard JavaBeans event model, and the standard Observer/Observable event model, where necessary due to legacy support of the evolving standard Core Java Class Library.

### 5.3.5 TAKO Combined Interface

The core knowledge management features of TAKODO are presented through the TAKO interface (Figure 48), which combines into a unified interface the linguistic, inferencing, message delivery, querying and filtering functions of the underlying TA, KO and other subsystems. The TAKO interface provides the API for using TAKODO as a knowledge server, independent of the user interface.

### 5.3.6 Representation of Data

Abstract interfaces specify the API for interacting with TAKODO objects. This includes the API (Figure 49) for knowledge base elements, including statements and parts thereof, such as belief values or lexemes as used for subjects, predicates, complements, etc. Collections of objects, such as the collection of statements in a knowledge base, conform to the standard Java Collections API, with extensions to the framework that add such features as the ability to query the collection to retrieve, for example, only those objects satisfying some Filter, as described below.

As discussed, documents and document fragments such as sentences, words, and arbitrary phrases have an object-oriented API (Figure 50). This API also defines the interfaces required by linguistic services such as stemming, lemmatization, and access to syntactic and part-of-speech tagging information.

### 5.3.7 Filters and Queries

TAKODO supports a wide array of filters, supporting string matching, regular expressions, word class restrictions and numerical restrictions. Different filters are implemented by different concrete classes. The design (Figure 51) is meant to be flexible, to allow the easy extension or replacement of existing filters.

Every filter must provide the implementation of a method to evaluate whether a given object matches the filter. This minimal implementation permits the use of a filter without any additional knowledge about its semantics, although such use requires the sequential testing of each information item in turn, resulting in performance that is linear in the number of information items. However, specialized support for specific filters may allow greater optimization. For example, the TAKODO interface to knowledge management systems accepts arbitrary filters to limit the set of statements for retrieval, but recognizes and thereupon optimizes access when used with instances of specific filter classes. In effect, filter instances become a means of communicating filter specifications, akin to a query language but with the ability to transmit the required computational behavior.

In TAKODO, a query may or may not be a filter. That is, queries may specify requirements that retrieved information items must satisfy, as well as additional requirements such as the needs for presentation or organization of retrieved information items. The former requirements are, in effect, a filter. The latter requirements may specify how the retrieved items are to be collated, for example.

*TAKOQuery*, the class used for representing TAKODO queries, is not a filter, but does use aggregation to contain instances of the *TAKOPattern* class. These instances are filters (in Java OOP terms, *TAKOPattern* implements the *Filter* interface) applicable to indexing terms, with additional semantics as patterns for searching within sentences.

Specifically, methods exist to find the start and end offsets of the pattern within regions of a given phrase. *TAKOPattern* defines several other methods allowing efficient use in the presence of indexing technology, such as a full-text inverted index. One of these methods may optionally return a term list, allowing retrieval to limit the scope to fragments containing the specified terms, which may then be tested for further conformance to any other restrictions of the pattern.

### 5.3.8 Communication with the Text Analyzer

The interface to TA services is handled by the TAKODO TA module, which is divided into linguistic services, statistics gathering, and question answering. Actual communication with the TA is via sockets using the TA server's protocol for requesting execution of the question answering system. Retrieval of results is performed directly from the resulting Oracle tables, as specified by the TA's API. This is the sole form of communication made available by the TA to external applications. All other TA-related functionality has either been re-implemented completely, or has been derived from TA tables in the database schema. In particular, this means that all concordance operations have been re-implemented in TAKODO by the author. As TA does not support query expansion or stemming, these are handled by TAKODO by sending multiple requests to the TA server.

### 5.3.9 Communication with a Knowledge Management System

The TAKODO KO module provides a generic, abstract interface to knowledge management systems, with concrete implementations for the Knowledge Organizer (KO) and its predecessor, DocKMan's Knowledge Base Builder (DKBB).

#### *5.3.9.1 Communication with DocKMan's Knowledge Base Builder (DKBB)*

Legacy support of DKBB knowledge bases is handled by the DocKMan component of the TAKODO KO module. As described, DKBB does not offer a public API for interacting with the knowledge base independently of the user interface, and even then, the API is incomplete for knowledge management purposes. Therefore, TAKODO directly manipulates the underlying Sybase database, and re-implements all required

knowledge manipulation and inferencing features, including a limited re-implementation of statement inheritance. DKBB's external API is used solely for event notification.

#### *5.3.9.2 Communication with the Knowledge Organizer*

The interface to the Knowledge Organizer (KO) is handled by the WJKO[75] component of the TAKODO KO module. Communication with KO employs its public API. As TAKODO and KO are both available for the Java 2 Platform and have a shared code base, no special considerations are required.

### 5.3.10 Communication with Other Information Sources

In TAKODO, information sources (Figure 52) are called Oracles, and are not to be confused with the Oracle RDBMS. Oracles are accessed through the TAKODO Oracle module. TAKODO Oracle defines a minimal interface needed for presenting information needs with subsequent retrieval of results. Since Oracles are Java beans, the addition of more information sources is trivial, requiring little more than registration with TAKODO's Oracle Manager. Several concrete implementations of the TAKODO Oracle interface provide access to knowledge bases, the answers from the TA, and the document base itself.

TAKODO uses an abstract interface (Figure 53) to communicate with knowledge management systems. This interface defines access to knowledge base meta-information, allows selection of knowledge bases, and provides means for interacting with the knowledge base to assert new knowledge or retrieve existing knowledge. Concrete implementations provide access to knowledge management systems such as the Knowledge Organizer (KO).

The author had considered using an existing standard interface to knowledge management systems, for greater interoperability. The choices included Open Knowledge Base Connectivity (OKBC) and its predecessor, the Generic Frame Protocol (GFP) [SRI International, 1995]. However, neither DKBB nor KO implement such standard interfaces. As these standards are designed to accommodate very general needs, their API

[75] During development, the acronym KO was used generically, to mean any knowledge management system, and the internal development name of the present Knowledge Organizer was WJKO.

is more complex than that needed by TAKODO. Consequently the implementation cost was evaluated as excessive for the research purposes. In any case, it is always possible to create a concrete implementation of the TAKODO interfaces that acts as a bridge to OKBC-compliant knowledge management systems, if ever the need arises for increased interoperability.

## 5.4 Information Retrieval Issues

### 5.4.1 Term Weighting

The system supports term weighting internally, such as during query expansion. This weighting exists primarily for the benefit of future relevance-feedback mechanisms. However, this term weighting capability is not presented in the user interface. The justification lies in the difficulty users may have in understanding the consequences of arbitrary manual weight reassignment [Yang et al., 1998].

The statistics on word usage, as computed during the creation of the inverted index, allow the use of conventional tf·idf term weights.

### 5.4.2 Query Expansion

At present, query expansion employs two sources. The first is the knowledge base, which stores the statements expressing thesaural relationships between concepts. The strength of the thesaural relationships is directly represented using the statement's belief weights.

The second source is the full-text inverted index, as described, which groups morphological variants under their lexeme. Hence, it is possible to request all words sharing a common word stem or lemmatized form.

### 5.4.3 Relevance Ranking of Sentences

Conceptually, the relevance of a sentence is computed as a weighted sum of the relevance scores associated with two information streams derived from the sentence. The first stream is the set of word-types occurring in the sentence, with relevance computed as similarity to query terms using the vector space model and dot product. The second stream is the set of subject-verb-object clauses embedded within the sentence, with

relevance computed using penalties for failures to match parts of the statement represented by the OAV model for queries.

### 5.4.4 Collection Fusion

TAKODO performs collection fusion by normalizing relevance scores. The present methodology is somewhat unsophisticated. Different information sources are associated with different weights, which are used as factors to adjust the relevance scores of the information items they provide. For research purposes, the weights have been determined experimentally. The collection fusion strategy corresponds to the *merge-sort* technique as described in [Towell et al., 1995]. For more general use with various heterogeneous information sources, a better collection fusion strategy is required. Towell et al. note that a poor collection fusion strategy may result in retrieval with significantly lower precision than retrieval from a single collection.

# Chapter 6: Conclusions

## 6.1 Summary

The purpose of this thesis was to create a software tool with a single, uniform user interface, designed to help ease several information management problems. Chief among these was the retrieval of specific facts, such as those needed to answer some question within a given domain.

The resolution of these problems consisted of two parts. The first was the provision of effective knowledge management tools to facilitate the creation of knowledge bases; these store their contents in a compact, knowledge-rich representation that is particularly useful for satisfying information needs. The second was the integration of information retrieval techniques, particularly passage retrieval, to provide the support of existing on-line documents in satisfying information needs. This is needed due to the limited availability of complete knowledge bases.

This thesis has shown that coupling explicit domain knowledge with a document base provides mutual benefits: retrieval is more "intelligent" and the information base is more complete.

A secondary goal was the creation of an extensible system, able to serve future needs and capable of accommodating additional subsystems, with seamless integration and limited impact to the existing code base. Lessons learned from previous prototypes showed that rapid applications development is useful for experimentation, but without adequate consideration of software maintainability, the resulting system may poorly satisfy further research needs.

Chapter 2 presented the needed conceptual background. Chapter 3 described the Text Analyzer and the Knowledge Organizer. Both of these were integrated into TAKODO, as presented in Chapter 4. Chapter 5 presented a brief and necessarily incomplete look at various implementation details.

## 6.2 Discussion

The main conclusion from this research is that it is indeed feasible to provide unified and simultaneous access to both knowledge bases and document bases. The former stores high-quality information, often with some formal semantics. The latter stores ambiguous, natural language texts, but may be far more complete than the knowledge base. The direct access to both benefits information retrieval tasks, and aids during knowledge engineering processes. Although the LAKE group has not completed any in-depth usability surveys, initial user feedback in the form of word-of-mouth is promising. The main points are that:

- Query interfaces, as used by search engines but with minor modifications, are an effective means for users to interrogate a knowledge base. In very large knowledge bases, navigation may be inappropriate when you know what you want, but don't know where to find it.

- There is a need for various degrees of formality in knowledge representation. Information retrieval systems target natural language text, which is at one extreme. Text presents major difficulties due to its ambiguity and complexity, allowing many ways of expressing the same idea. Formal knowledge management systems typically target the other extreme, often requiring their content to be absolutely correct, consistent, and obeying strict formal semantics. TAKODO imposes some minimal requirements, with semantics associated with the subject, predicate, and complement parts of statements, and recommends the use of ClearTalk conventions. However, TAKODO also allows statements to contain arbitrary unprocessed text (for example, informal English descriptions, or even computer code). Certainty factors are used to allow the modeling of beliefs, which may be contradictory. Like CODE4 and its successors, TAKODO offers such formality, but does not require it.

- It is believed that cooperative knowledge acquisition should be made more accessible to a larger class of users. Of course, the information entered by users without formal knowledge engineering skills may not be of the same quality as

that entered by experts, but it remains nonetheless useful. An individual may have a specific question and use an information retrieval system to find its answer. Thereupon, the query results are lost to other users who may have the same question. The current state of the art remains in the form of Frequently Asked Questions (FAQ) lists. TAKODO, instead, allows the user to store retrieved information into a lightly structured knowledge base, which may be refined subsequently.

Issues listed as future work during prior projects by the LAKE group [Kavanagh, 1995; Zayour, 1997; Prpic, 1997a] and which have been implemented by the author during the development of TAKODO include:

- Transparent querying of multiple information sources with fusion of the results, which includes: (i) information retrieval from both the knowledge base and document base; and (ii) complete and transparent integration of the TA with the knowledge management system.

- Provisions for rapid construction of a knowledge base through: (i) initial experimentation with automation of the task for the construction of a knowledge base from the document base; (ii) continuous availability of a lexicon in the form of a sorted term list; (iii) ability to find relevant predicates associated with a given subject from the knowledge base, as candidates for new statements during knowledge entry; (iv) extensive copy/paste functionality in the user interface, for increased efficiency during manual knowledge entry from the document base; and (v) automatic maintenance of audit trails for tracing knowledge back to its sources.

- Use of concurrency with incremental retrieval to minimize the impact of delays during query processing.

- Linguistic processing (stopwording, lemmatization, stemming) made available to all systems, including the TA and the KO, to resolve issues of morphological variation and multiple inflected word forms.

- Greater flexibility and near universality for filtering and sorting of all kinds of information items; support for complex pattern matching using regular expressions;
- Support for multiple kinds of taxonomies (hyponymic, meronymic) including topic hierarchies.
- Presentation of formatted text and multimedia content (through HTML).

The greatest difficulties encountered during the implementation of TAKODO were related to two issues: (i) efficiency concerns, due to the interactive nature of the system; and (ii) the very rapid evolution of the technologies used during development, particularly of the Java Platform.

## 6.3 Future Work

There is much work left to complete the DocKMan project's vision of a unified system to integrate documents and knowledge bases, complete with any necessary linguistic knowledge.

1. A more complete usability evaluation is certainly necessary. Such a study should focus on retrieval effectiveness, with standard measures of precision and recall, as well as knowledge acquisition efficiency. The HPKB project [HPKB, 1999] provides a guide for the methodology of such a study.
2. Additional knowledge visualization techniques would be useful. Outline views, as in the Knowledge Organizer (KO), may yet be added for displaying and manipulating taxonomies. The graphical view of KO, presenting information as a semantic network, still has its applicability, especially when restricting the view to a small subgraph of the entire information base. Fisheye views were cited as future work for KO's predecessor, DKBB, but were only partially implemented. CODE4's property matrix browsers were a useful display for comparing subjects.

3. The *Related* pane should be enhanced to show answers that are related in several different ways, perhaps borrowing ideas from the WAVE Conceptual Browser [Kent & Neuss, 1997].

4. A query history function should be added, to maintain the sequence of queries used during an information retrieval session. A mechanism to persistently store the query for subsequent use may be equally beneficial.

5. Relevance feedback is very limited and rather unsophisticated in the current implementation. Any relevance feedback is used only to make additions to the knowledge base. It does not result in any automatic query reformulation.

6. It may be possible to improve and generalize query expansion beyond thesaural relationships and morphological variations, by employing spreading activation on the semantic network represented by the knowledge base. Such spreading activation may yield improved concept recognition abilities, by taking into account the shared context in which the original query terms are used.

7. TAKODO does not yet provide explicit support for KO's concept of compound statements, and lacks the linguistically motivated *verb*, *verb prefix*, and *verb postfix* parts of a statement. These features were added to the KO late in the development process of the thesis.

8. Updating the document base with changes to existing documents is problematic: existing references to passages must be maintained in some way. The current strategy requires marking up the original documents with tags to store unique sentence identifiers. These must be left in place during any subsequent editing.

9. TAKODO needs to provide a simple and complete interface for document base administration. At present, this task is the responsibility of the TA, and the addition of new documents is excessively complex. Unfortunately, the TA does not provide an API to allow integration of the task into TAKODO. However, TAKODO does include administrative features to selectively re-index documents

following any updates. Ideally, one would like the information gathering facilities such as those of the Web Information Collector (WIC) [Zayour, 1997].

10. One may wish to replace the indexing technology with a more sophisticated implementation. Ideally, TAKODO would use an off-the-shelf IR system, but there is need of an open system to allow special customizations. For performance and scalability reasons, the TA should be improved to make use of full-text indexing. The options are currently under investigation by the LAKE group.

11. Additional options for importing and exporting the knowledge base are needed, with preference for standard formats, such as KIF or OML/CKML [Kent & Neuss, 1997]. KO can already export its knowledge base as an XML document, using its own schema. In particular, one may wish to store knowledge base statements directly in the source documents, as additional markup such as that of SHOE [Parallel Understanding Systems Group, 1999a]. Such a solution guarantees that the associated formal knowledge is always transported with the document.

12. True multi-user execution of TAKODO remains limited by the TA's single-client restrictions, requiring a dedicated instance of the TA server to be started for each user. At present, this limits the practicality of wide-area Web deployment.

13. With respect to shared knowledge bases, there is a need for shared ontologies, or at least, for the mapping between different ontologies. This is a difficult problem [Lenat & Guha, 1989; Lenat, 1995; Neches at al., 1991], inadequately addressed by this research.

In closing, let us make a final comment on the integration between TAKODO and KO. There is much overlap in the two systems. Both share some common formalisms using statements to represent information. Both display these statements as grids with several columns (although KO's columns are fixed, whereas TAKODO's depend dynamically on the table's contents). Both provide their own, independent but similar interfaces for modifying the knowledge base. During development, features have migrated in both directions. Although they satisfy different needs, and are expected to

diverge in some areas, TAKODO has progressively become more KO-like from its information retrieval origins, improving its knowledge management facilities, and KO has progressively become more TAKODO-like, improving its respective information retrieval facilities. Both systems are attacking the DocKMan project's goal of integrating documents with knowledge bases, but from different directions.

As research continues in the next decade, the author foresees increased integration of AI technologies with those of information retrieval. As users become increasingly inundated with the mass of available information, they may learn to appreciate the benefits of increased knowledge sharing resulting from greater formality in knowledge representation. At present, word processors are omnipresent and, despite their numerous features and significant complexity, are well understood by typical computer users; unfortunately, knowledge management systems are not. The author believes this will change, slowly. Documents are more than bags of words; they carry meaning. Natural language processing techniques, coupled with rich lexical databases and large knowledge bases, will hopefully allow retrieval of information that is less sensitive to the language and terminology used in natural language documents. However, this is no excuse for carelessly and *a priori* disregarding the option of contributing to a knowledge base in favor of writing a natural language document (that may be later misunderstood, even by other, human readers). The world needs better tools for knowledge management, including its acquisition and its retrieval, aiding authors to be clear and concise, use appropriate terminology, and avoid unnecessary duplication of effort.

# Appendix A: Filters

| Filter Syntax | Description |
|---|---|
| **Prefix Operator (First non-whitespace character): Optional, default is &** | |
| `&` | AND: Require matches of this filter, along with other filters. |
| `+` | OR: Include matches of this filter, regardless of any other filters. |
| `-` | NOT: Exclude matches of this filter, regardless of any other filters. |
| **Filter Specifier (Immediately following the Prefix Operator)** | |
| `\`*string* | Case insensitive substring match on *string*, including any leading or trailing whitespace. |
| `'`*string* | Exact string match on *string*, including any leading or trailing whitespace. |
| `"`*string* | Case insensitive exact string match on *string*, including any leading or trailing whitespace. |
| `/`*regex* | Perl-style Regular Expression match on *regex*, including any leading or trailing whitespace. |
| `op` *number* where `op` is one of `> < >= <= = <>` | Numeric comparison. Strings are converted to numbers, or treated as null if not valid. Can also search for nulls using = without any value. |
| `@engcg:`*data* | Match words on the ENGCG base form specified by the *data* (the word must be tagged) |
| `@stem:`*data* | Match words on the stemmed form specified by the *data* (the word must be tagged). |
| `@lemma:`*data* | Match words on the ENGCG Stemmed lemma specified by the *data* (the word must be tagged). |
| `@root:`*data* | Match words on any of the ENGCG Stemmed lemmas found to exist for the word specified by the *data* (the word must be tagged). |
| `@same:`*data* | The filter is expanded to match on all words having a common ENGCG Stemmed lemma with the word specified by the *data*. |
| `@class:`*include/exclude* | An item must match at least one of the specifiers in the inclusion list (logical OR), but must not match any of the items in the optional exclusion list (logical AND). The inclusion list specifies, in any order, the tags and classes among those of Table 19. |
| `@`*relation*`:`*data* | Match on lexemes related to the one specified, using the given relation type. *relation* may be one of:<br>• `isa`: Includes all broader terms (hypernyms)<br>• `sub`: Includes all narrower terms (hyponyms)<br>• `syn`: Includes all synonyms<br>• `related`: Includes all related forms (hypernyms, hyponyms, synonyms, conjugations)<br>The optional *prefix* may be one of: |

| | |
|---|---|
| | • ': Exact match.<br>• ": Case insensitive match.<br>• \ (default): Case insensitive substring match. |
| ? | *Reserved*: Expression which evaluates to true/false. # is the object being searched for. |
| ~ | *Reserved*: String match. Following char may be one of = for exact match, * for substring, and \ for case-insensitive. |
| :*syntax*: | *Reserved*: Other syntax |
| Any other specifier starting with a different first character than the above list | Case insensitive substring match, excluding trailing whitespace. |

*Table 17: Filter Syntax in TAKODO*

The following stoplist was derived from the Text Analyzer stoplist [Kavanagh, 1995] and from the Brown corpus[76], with further additions during experimentation.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1988 | being | except | into | not | sincere | up |
| 1989 | below | f | is | nothing | six | upon |
| 1990 | beside | false | isn't | Nov | sixty | us |
| 1991 | besides | Feb | it | November | so | use |
| 1992 | between | February | it_ | now | some | used |
| 1993 | beyond | few | its | nowhere | somehow | uses |
| 1994 | bill | fifteen | it's | o | someone | using |
| 1995 | both | fify | itself | Oct | someone=else | v |
| 1996 | bottom | fig | j | October | something | very |
| 1997 | but | figure | Jan | of | sometime | via |
| 1998 | by | fill | January | off | sometimes | w |
| 1999 | c | find | July | often | somewhere | was |
| 2000 | call | fire | June | on | still | we |
| a | can | first | k | once | such | week |
| a=few | cannot | five | keep | one | system | weeks |
| about | cant | for | know | one=another | t | well |
| above | can't | former | l | one=or=more | take | were |
| across | co | formerly | last | only | ten | what |
| after | computer | forty | latter | onto | than | what_ |
| afterwards | con | found | latterly | or | that | whatever |
| again | could | four | least | other | that_ | when |
| against | couldnt | from | less | others | the | whence |
| all | couldn't | front | like | otherwise | their | whenever |
| almost | cry | full | ltd | our | them | where |
| alone | d | further | m | ours | themselves | whereafter |
| along | dd | g | made | ourselves | then | whereas |
| already | de | get | make | out | thence | whereby |
| also | Dec | give | man | over | there | wherein |
| although | December | go | many | own | thereafter | whereupon |

[76] Stoplist available from anonymous source: http://local.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/

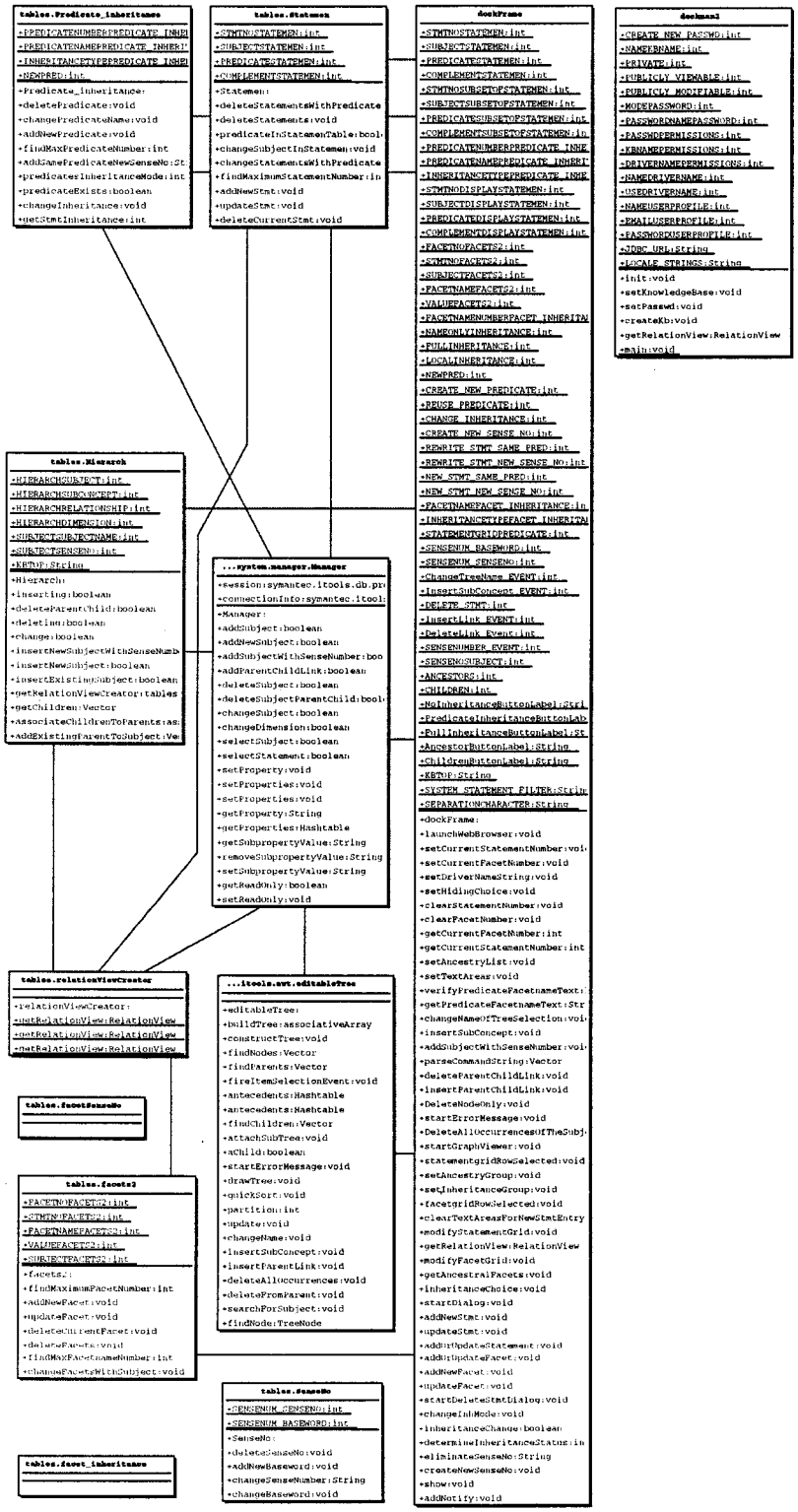
| always | describe | h | Mar | p | thereby | wherever |
|---|---|---|---|---|---|---|
| am | detail | H1 | March | part | therefore | whether |
| among | did | h2 | May | per | therein | which |
| amongst | didn | had | me | perhaps | thereupon | while |
| amoungst | dl | has | meanwhile | please | these | whither |
| amount | do | hasnt | might | put | they | who |
| an | does | have | mill | q | they'll | who_ |
| and | doesn't | having | mine | r | thick | whoever |
| another | done | he | month | rather | thin | whole |
| any | don't | hence | months | re | third | whom |
| anyhow | down | her | more | Record | this | whose |
| anyone | dt | here | moreover | s | this=one | why |
| anything | due | hereafter | most | said | those | will |
| anyway | during | hereby | mostly | same | though | with |
| anywhere | e.g | herein | move | say | three | within |
| April | e | hereupon | much | says | through | without |
| are | each | hers | must | second | throughout | would |
| around | each=other | herself | my | see | thru | x |
| as | eg | him | myself | seem | thus | y |
| at | eight | himself | n | seemed | to | year |
| Aug | either | his | name | seeming | together | years |
| August | eleven | how | namely | seems | too | yet |
| b | else | however | neither | Select | top | you |
| back | elsewhere | hundred | never | Sept | toward | you_ |
| be | empty | I | nevertheless | September | towards | you'll |
| became | enough | i.e | new | serious | true | your |
| because | et_al | i_ | next | seven | twelve | you're |
| become | etc | i | nine | several | twenty | yours |
| becomes | even | ie | no | she | two | yourself |
| becoming | ever | if | no_one | should | two-thirds | yourselves |
| been | every | in | nobody | shouldn | u | z |
| before | everyone | inc | none | show | un | |
| beforehand | everything | indeed | noone | side | under | |
| behind | everywhere | interest | nor | since | until | |

*Table 18: Default Stoplist Used in TAKODO*

| Class Type | Word Class | TAKODO Notation | TA Notation |
|---|---|---|---|
| **Importance** | Term | + | *None* |
| | Stop Word | - | *None* |
| | Content Word | * | *None* |
| **Syntactic Tags** | Auxiliary Verb | x | 'X |
| | Main Verb | v | 'V |
| | Subject | s | 'S |
| | Formal Subject | f | 'F |
| | Direct Object | o | 'O |
| | Indirect Object | i | 'I |
| | Pcompl-S | p | 'P |
| | Pcompl-O | c | 'p |
| | Adverbial | a | 'A |
| | Determiner | d | 'D |
| **Part of Speech Tags** | Adjective | A | ^A |
| | Abbreviation | B | ^B |
| | Adverb | D | ^D |
| | Coordinating Conjunction | & | ^& |
| | Subordinating Conjunction | C | ^C |
| | Determiner | T | ^T |
| | Infinitive | I | ^I |
| | Noun (Singular) | S | ^S |
| | Noun (Plural) | L | ^L |
| | Noun (Genitive) | G | ^G |
| | Noun | N | ^N |
| | Number | # | ^# |
| | PCP1 -ing | 1 | ^1 |
| | PCP2 -ed | 2 | ^2 |
| | Preposition | R | ^R |
| | Pronoun | P | ^P |
| | Verb | V | ^V |
| **Word Groups** | Part of a Compound | < | *None* (Compound treated as whole) |

*Table 19: Word Classes (Tags) in TAKODO*

# Appendix B: UML Class Diagrams




*DocKMan's Knowledge Base Builder*

*Figure 42: Abridged Class Diagram of DocKMan's Knowledge Base Builder API*

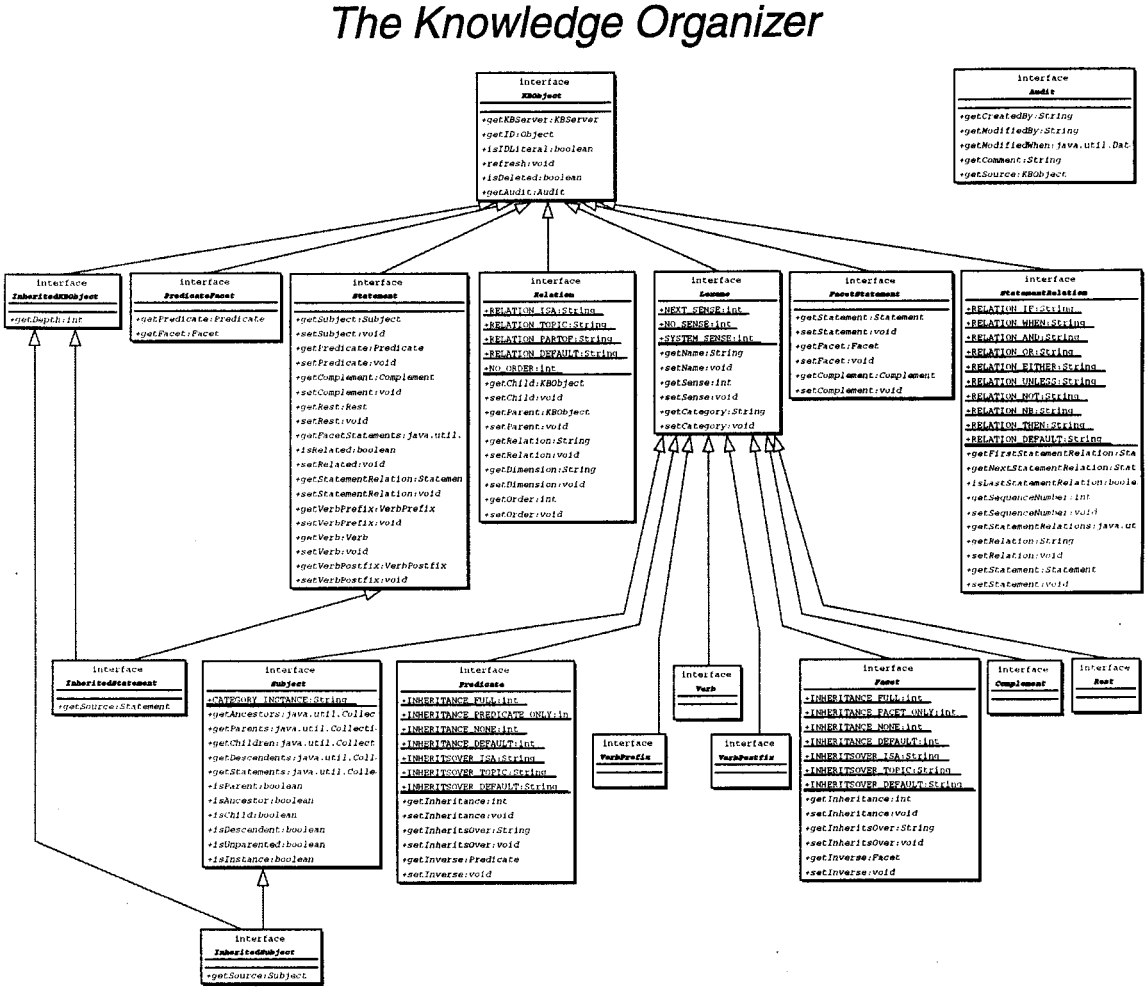



*Figure 43: Class Diagram of KO Knowledge Base Elements API (Abstract Interfaces)*

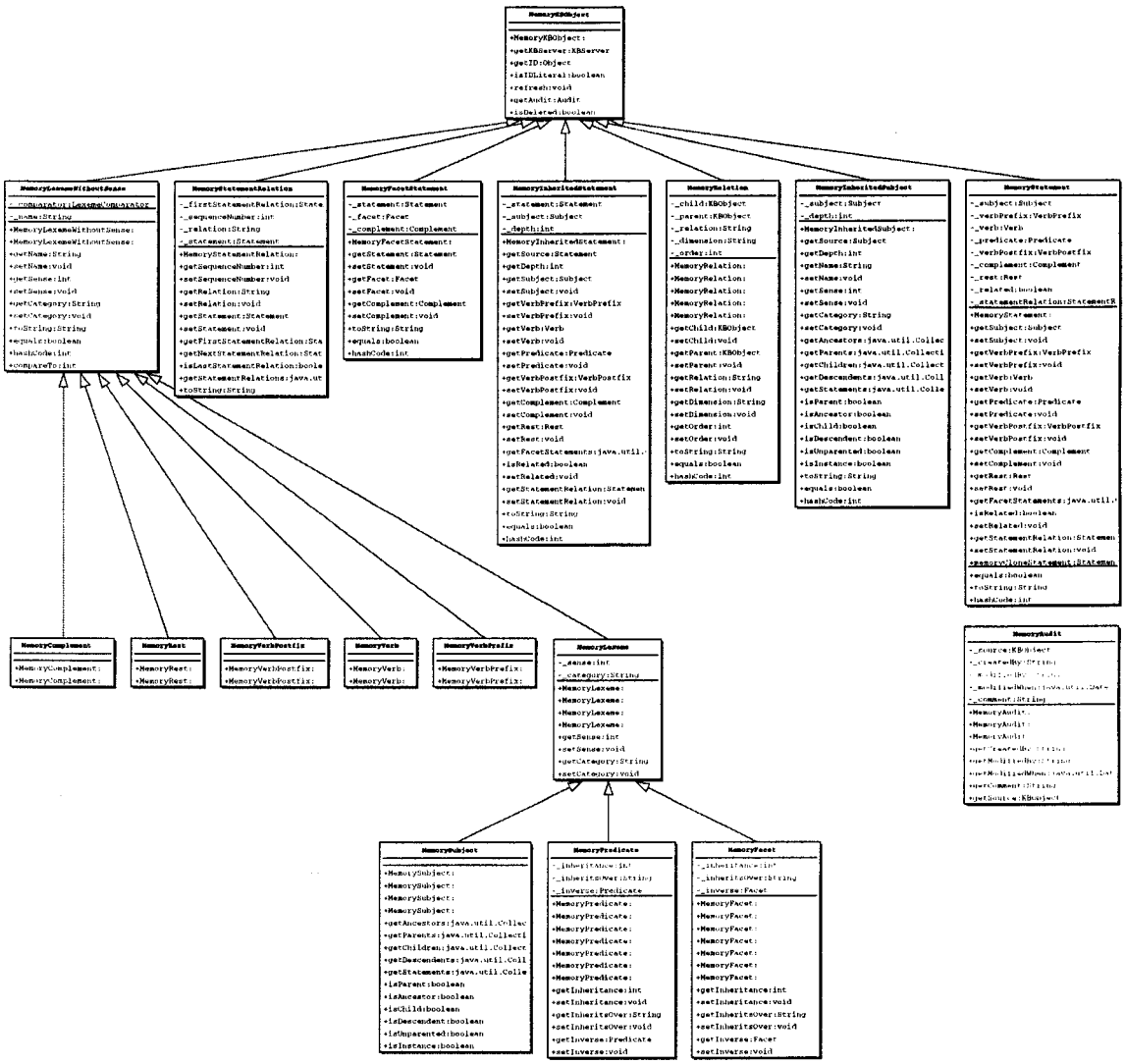


*Figure 44: Abridged Class Diagram of KO In-Memory Non-Persistent Knowledge Base Elements*

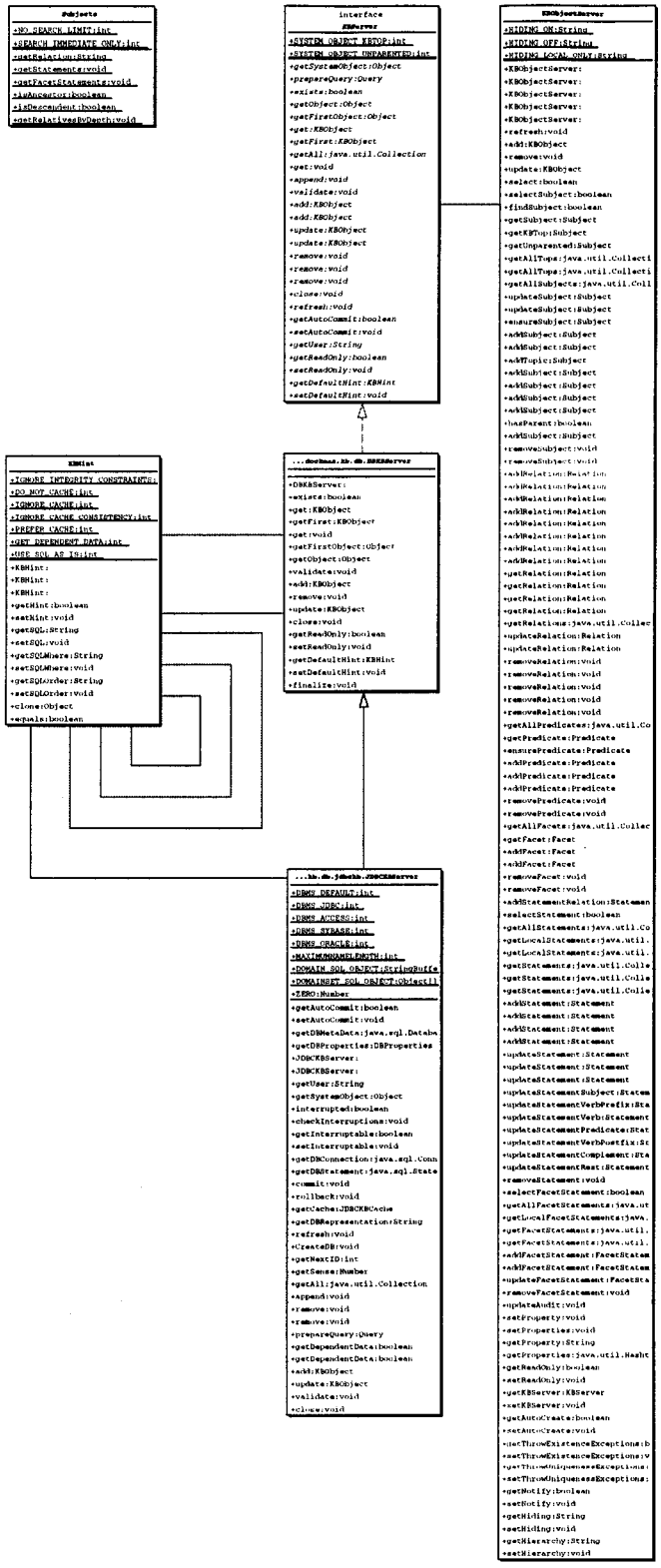


*Figure 45: Abridged Class Diagram of KO Server API*

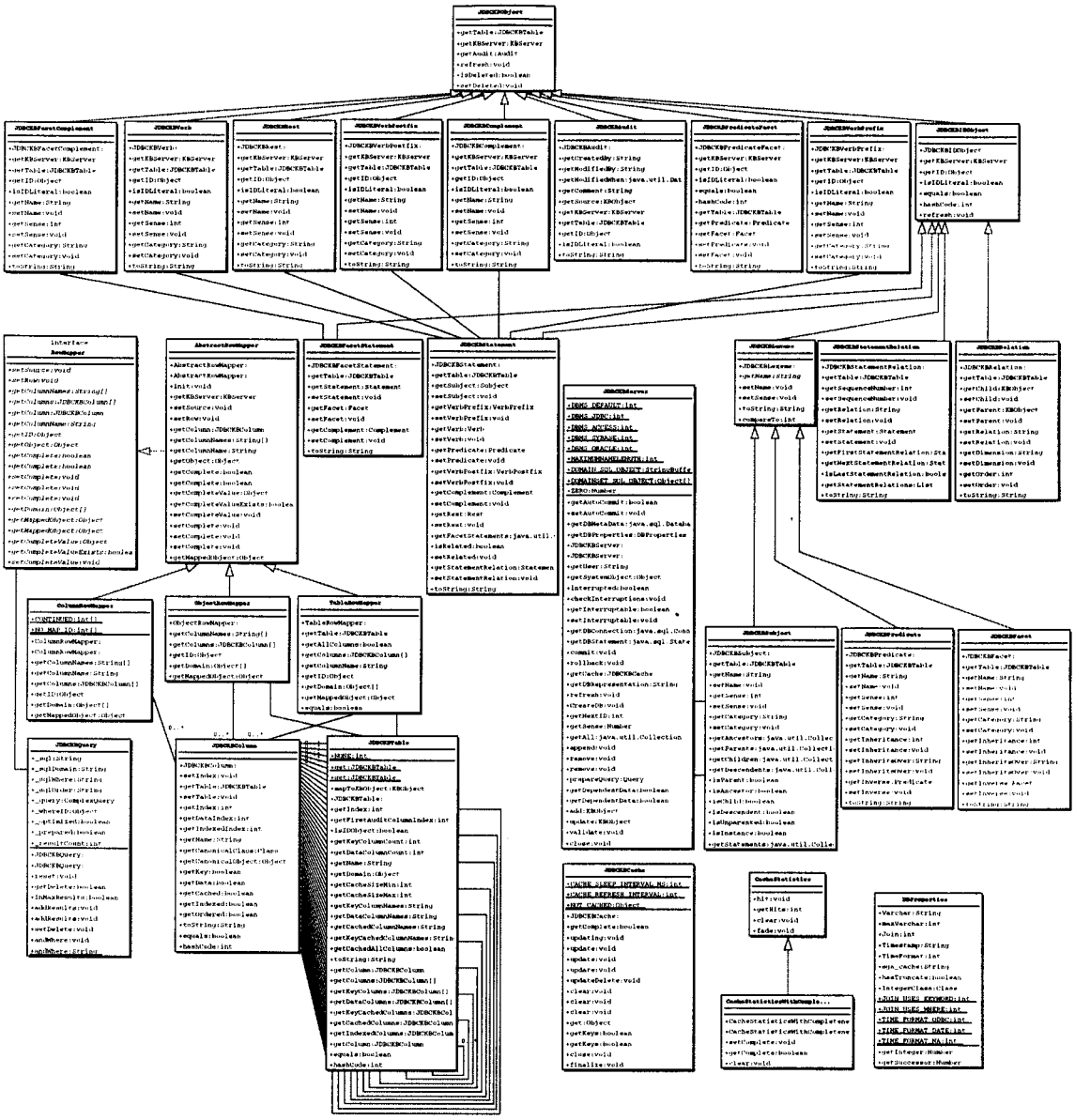


*Figure 46: Abridged Class Diagram of KO Server Implementation*

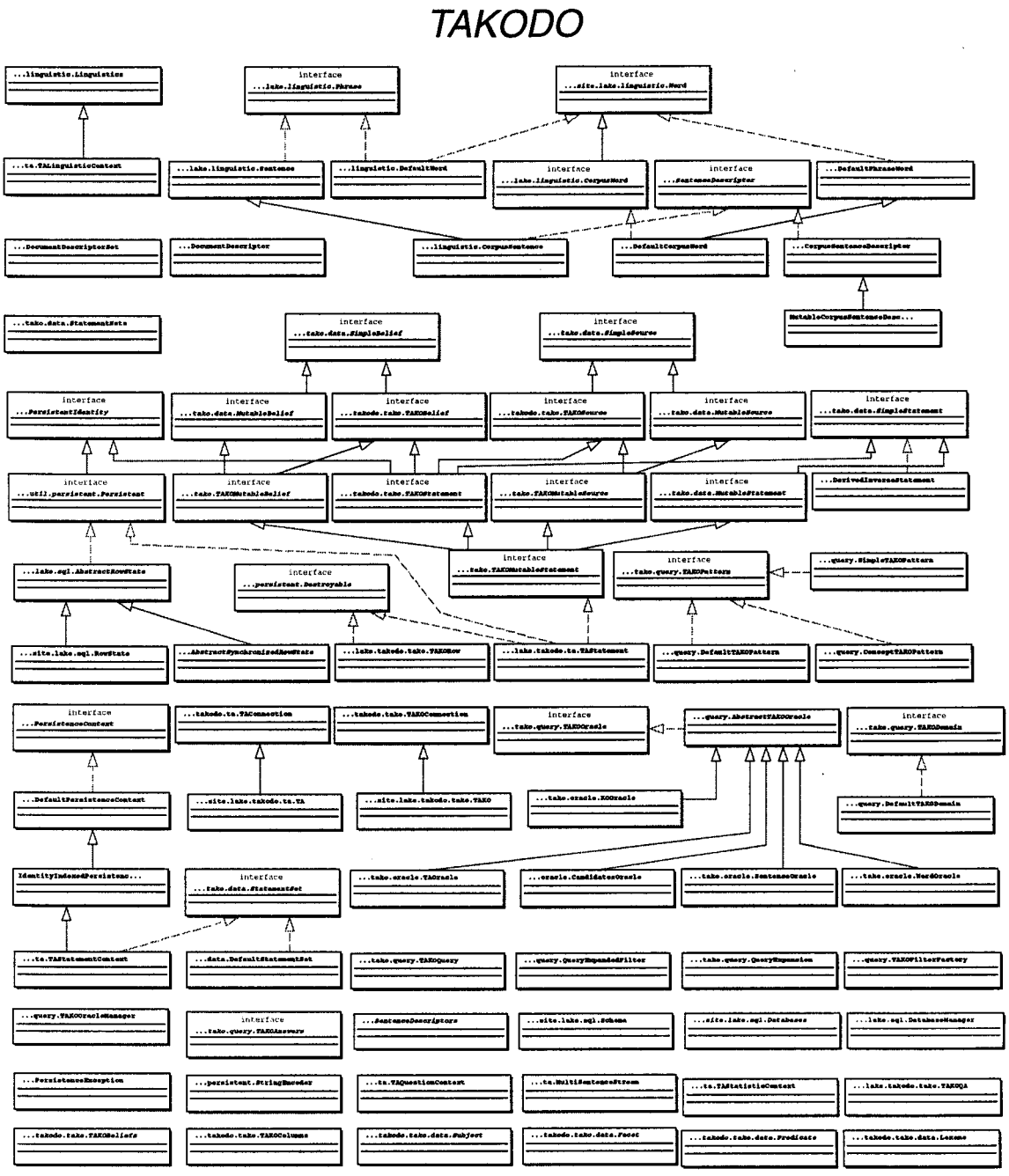



*Figure 47: Abridged Class Diagram of Main TAKODO Classes*

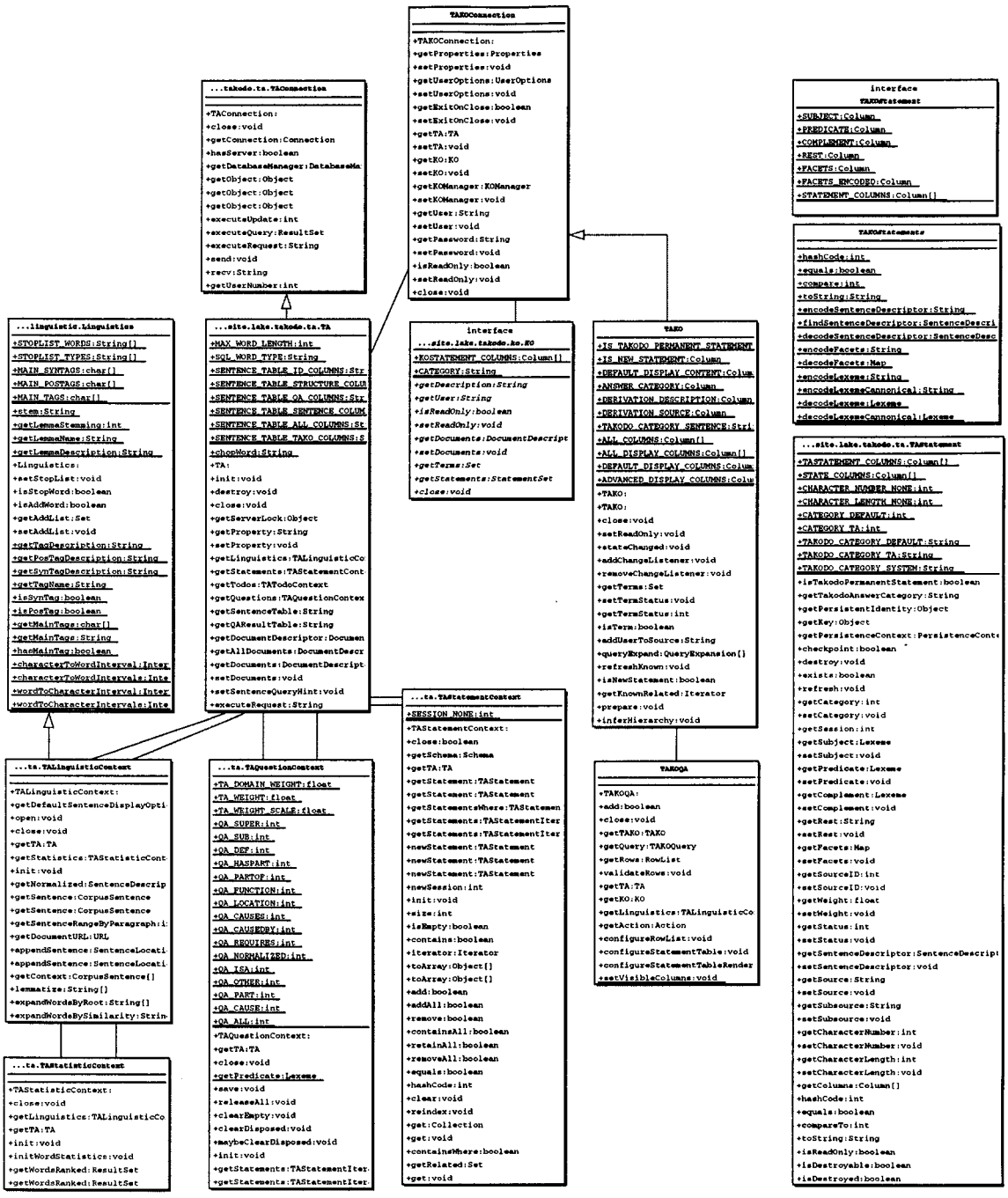



*Figure 48: Abridged Class Diagram of TAKO Server*

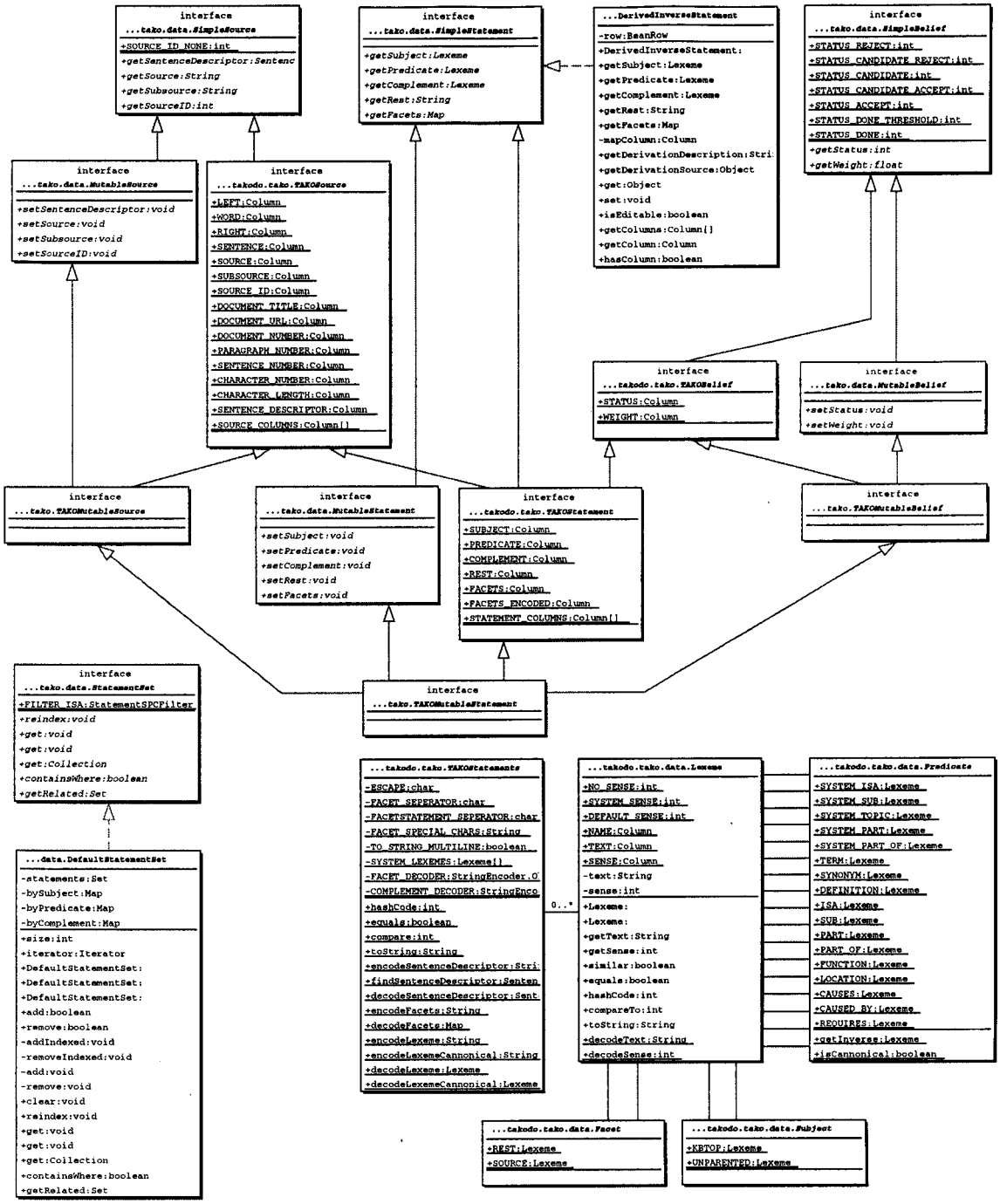



*Figure 49: Abridged Class Diagram of TAKODO Data Objects, Including Knowledge Base Elements*

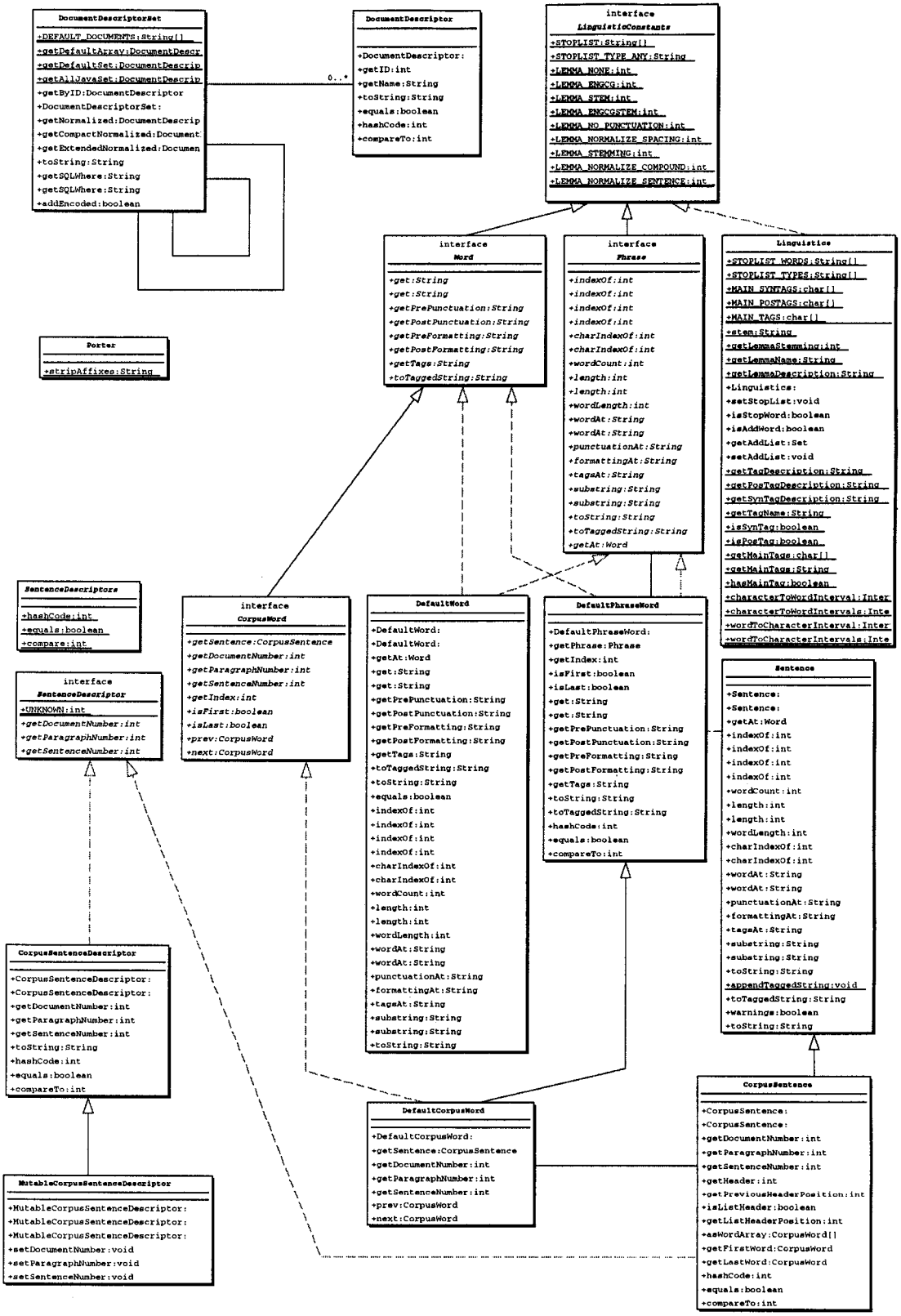



*Figure 50: Abridged Class Diagram of TAKODO Linguistics*

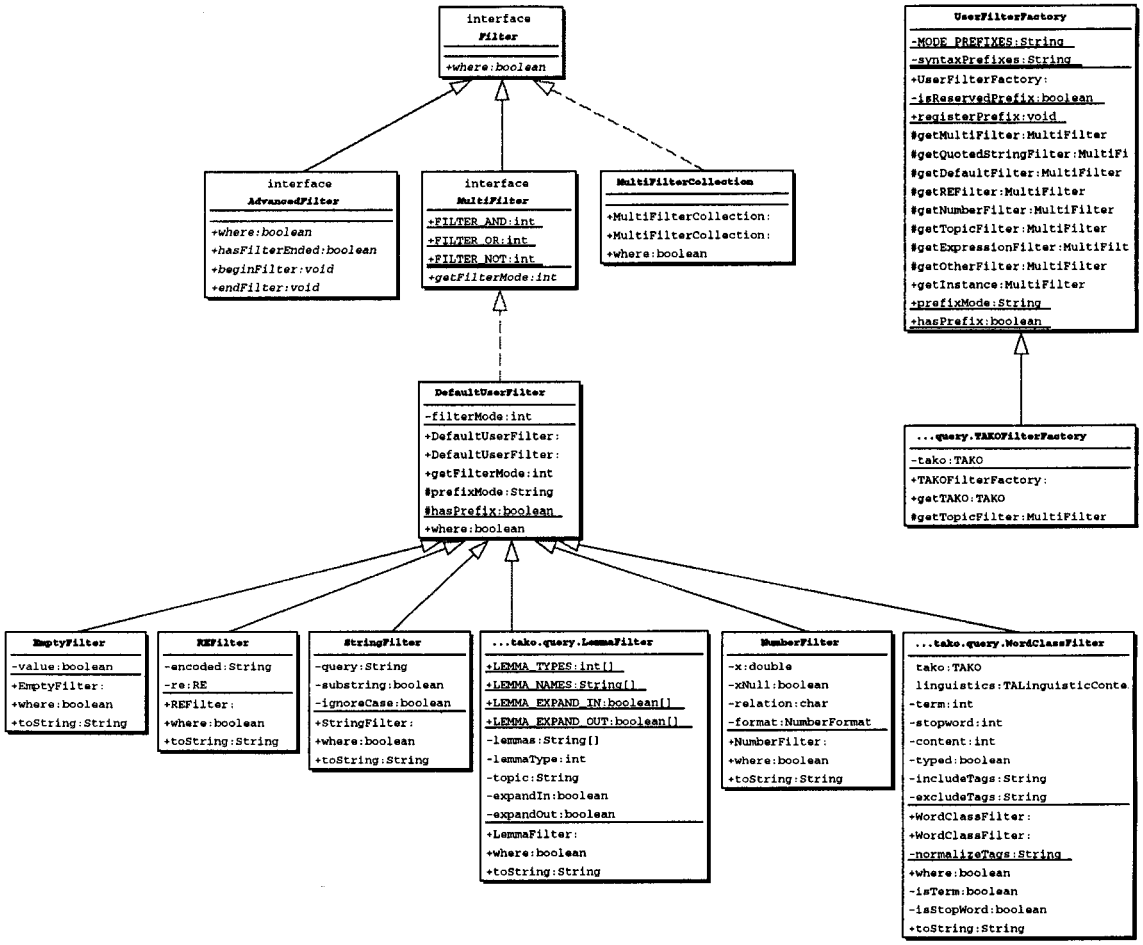



*Figure 51: Class Diagram of TAKODO Filters*

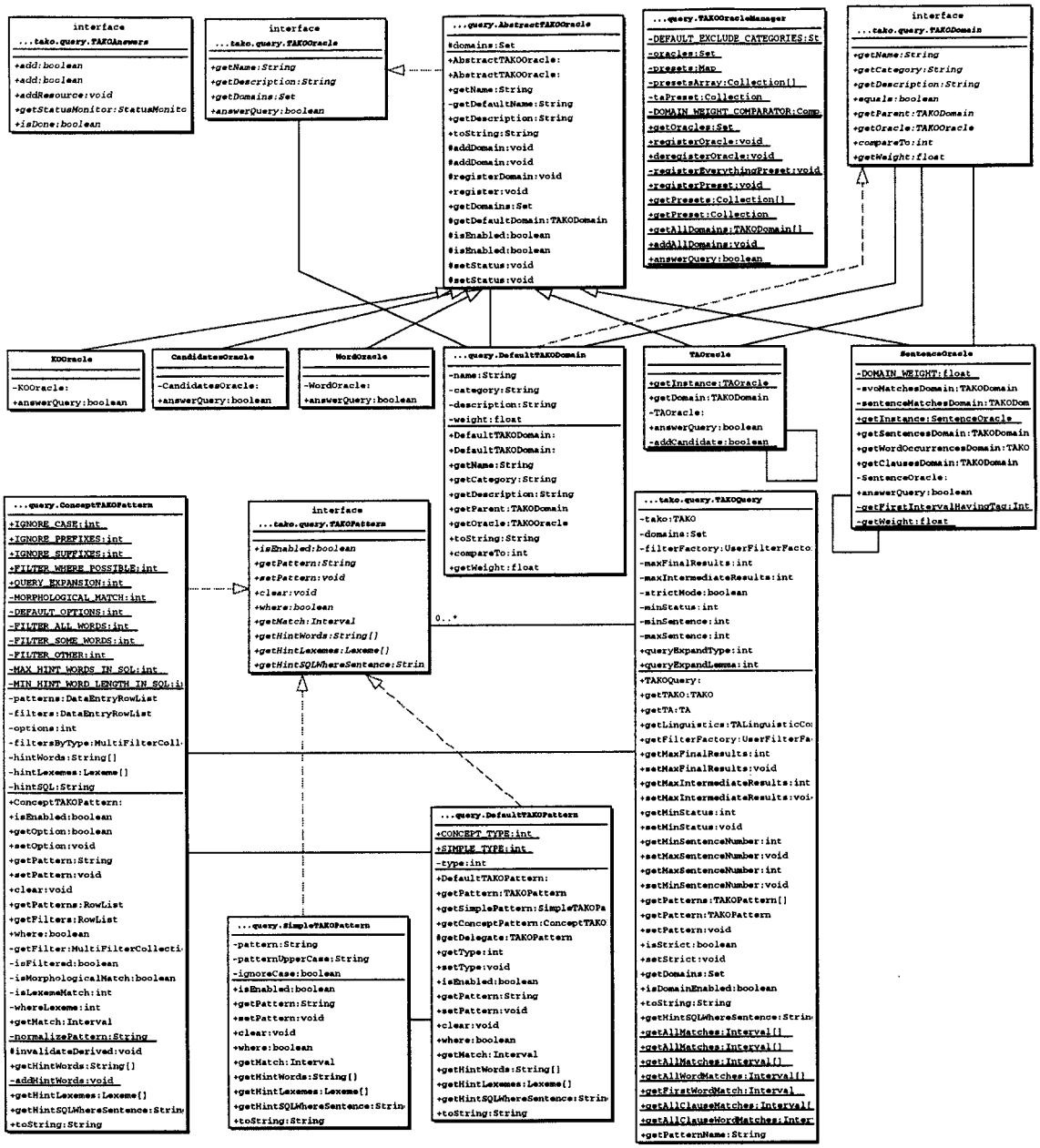



*Figure 52: Class Diagram of TAKODO Oracles API*

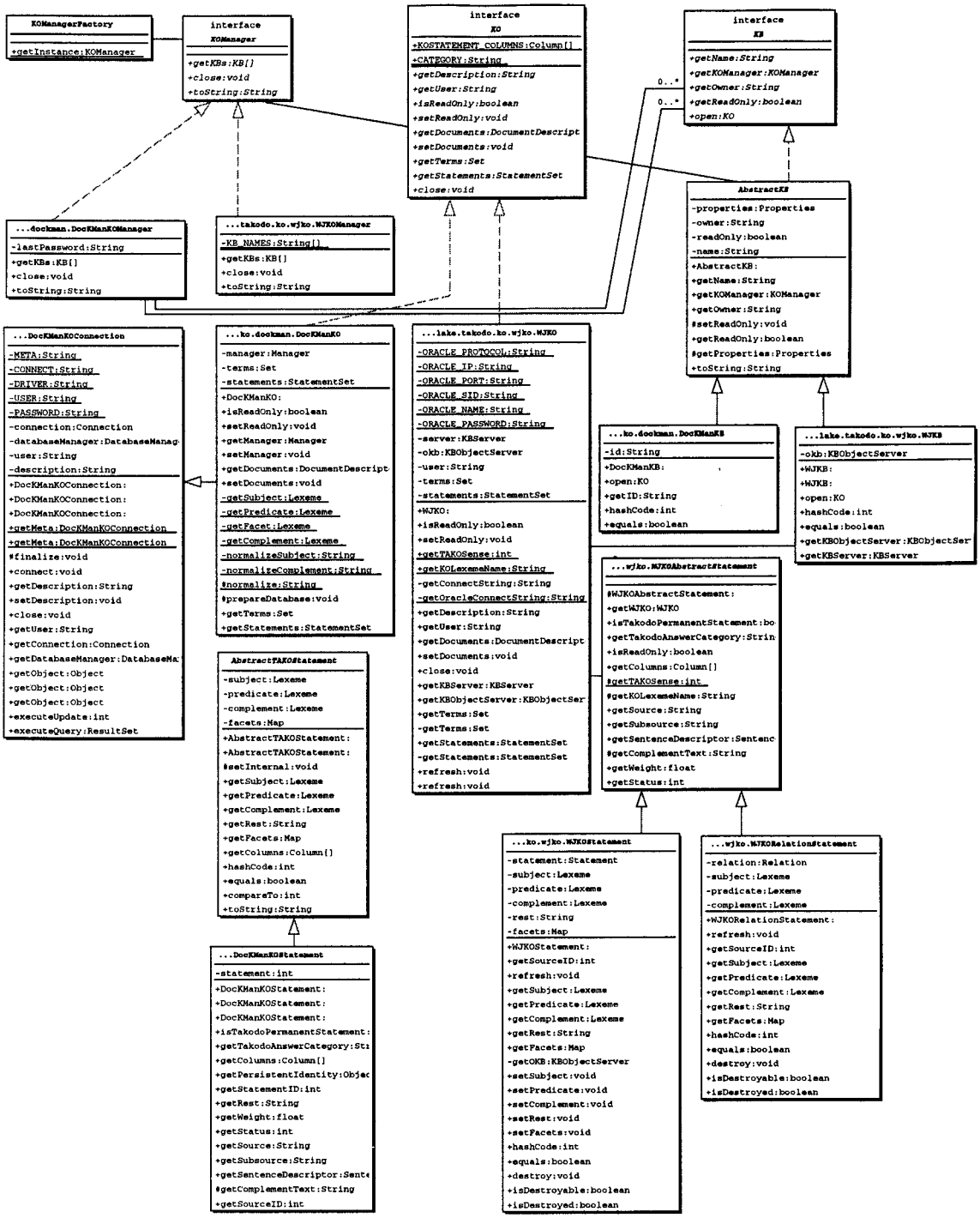



*Figure 53: Abridged Class Diagram of TAKODO Knowledge Management System API and Concrete Implementations*

# Bibliography


**AMBLER, S. W.** (1997). *Mapping Objects to Relational Databases.* [Online] Available: http://www.ambysoft.com/mappingObjects.pdf

**AMBROZIAK, J. & WOODS, W. A.** (1998). Natural Language Technology in Precision Content Retrieval. [Online] The SML Technical Report Series Technical Report TR-98-69. In *Proceedings of the International Conference on Natural Language Processing and Industrial Applications*, August 18-21.

**AUSSEN-GILLES, N., BOURIGAULT, D., CONDAMINES, A. & GROS, C.** (1995). "How Can Knowledge Acquisition Benefit from Terminology?" In *9th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff,.

**BAEZA-YATES, R.** (1992). Introduction to Data Structures and Algorithms Related to Information Retrieval. In Frakes & Baeza-Yates, 1992a (pp. 13-27).

**BAEZA-YATES, R. & RIBEIRO-NETO, B.** (1999). *Modern Information Retrieval.* New York: ACM Press.

**BARNBROOK, G.** (1996). *Language and Computers: A Practical Introduction to the Computer Analysis of Language.* Edinburgh: Edinburgh University Press.

**BELKIN, N. J., & CROFT, W. B.** (1987). Retrieval Techniques. In Williams M. (Ed.), *Annual Review of Information Science and Technology* (pp. 109-145). New York: Elsevier Science Publishers.

**BELL, G.** (1995). *Knowledge Modelling & Adaptive Hypertext.* [Online] Department of Psychology, University of Nottingham. Available: http://www.psyc.nott.ac.uk/~grb/papers/pgconf-paper.html

**BENSON, M.** (1990). Collocations and general-purpose dictionaries.*International Journal of Lexicography, 3-1 (pp. 23-35).*

**BERRY, M. W.** (1996). Low-Rank Orthogonal Decompositions for Information Retrieval Applications. Technical Report CS-95-2284. In *Numerical Linear Algebra with Applications*, Vol. 1(1), 1-27.

**BERRY, M. W., DUMAIS, S. T., O'BRIEN, G. W.** (1994). *Using Linear Algebra for Intelligent Information Retrieval.* Technical Report CS-94-270. SIAM Review.

**BERNERS-LEE, T., FIELDING, R., & MASINTER, L.** (1998). *Uniform Resource Identifiers (URI): Generic Syntax.* [Online] The Internet Society RFC2396. Available: http://www.ietf.org/rfc/rfc2396.txt

**BOOCH, G.** (1993). *Object-Oriented Analysis and Design with Applications.* (2nd ed.) Redwood City: Benjamin Cummings.

**BROOKS, T. A.** (1997). *Orthography as a Fundamental Impediment to Online Information Retrieval.* [Online] Available: http://faculty.washington.edu/tabrooks/Documents/Ortho3.html

**BURKE, R., HAMMOND, K., KULYUKIN, V., LYTINEN, S., TOMURO, N. & SCHOENBERG, S.** (1997). Question Answering from Frequently Asked Question Files: Experiences with the FAQ Finder System. In *AI Magazine*, Vol.18, no.2 (pp. 57-66).

**CARPINETO, C., DE MORI, R., ROMANO, G.** (1998). Informative Term Selection for Automatic Query Expansion. [Online] In *Proceeding of the 7th Annual Text REtrieval Conference (TREC-7).* Available: http://trec.nist.gov/

**CHURCH, K. W. & MERCER, R. L.** (1993). Introduction to the Special Issue on Computational Linguistics Using Large Corpora. In *Computational Linguistics 19.1* (pp. 1-23).

**CLARK, J. & DEROSE, S.** (Eds.) (1999). *XML Path Language (XPath).* [Online] World Wide Web Consortium. Available: http://www.w3.org/TR/xpath

**COTTMAN, B.** (1998). *Componentware:Component Software For The Enterprise.* [Online] I-Kinetics. Available: http://www.i-kinetics.com/wp/cwvision/CWVision.htm

**CRESTANI, F.** (1995). *Applications of Spreading Activation Techniques in Information Retrieval.* Department of Computer Science, University of Padova, Italy.

**CROFT, W. B.** (1995, November). *What Do People Want from Information Retrieval?* [Online] D-Lib Magazine. Available: http://www.dlib.org/dlib/november95/11croft.html

**CRYSTAL, D.** (1997). *The Cambridge Encyclopedia of the English Language.* Cambridge: Cambridge University Press.

**CYCORP.** (1999). Cycorp: Makers of the Cyc Knowledge Server for artificial intelligence-based Common Sense. [Online] Available: http://www.cyc.com/.

**DAVIS, I.** (1996). *Adding Structured Text to SQL/MM Part 2: Full Text.* University of Waterloo, Waterloo, Ontario, Canada. Available: http://solo.uwaterloo.ca/trdbms/sqlmm/lhr24r1.rest.html

**DELUGACH** (Web maintainer) (1999). *Conceptual Structures Home Page.* [Online] Available: http://www.cs.uah.edu/~delugach/CG/

**DUNNING, T.** (1993). Accurate Methods for the Statistics of Surprise and Coincidence. In *Computational Linguistics 19.1* (pp. 61-74).

**EGYHAZY, C., PLUNKETT, T. K. & THOMPSON, D. M.** (1997). Intelligent Web Search Agents. [Online]. Available: http://csgrad.cs.vt.edu/~tplunket/article.html

**ETZIONI, O.** (1996). The World Wide Web: Quagmire or Gold Mine? *Special Issue of the Communications of the ACM,* November, Vol. 39, no. 11.

**FEDERAL NETWORKING COUNCIL** (1995). *Definition of "Internet".* [Online] Author. Available: http://www.fnc.gov/Internet_res.html

**FOLTZ, P.W.** (1996) Comprehension, Coherence and Strategies in Hpyertext and Linear Text. In Rouet, J.-F., Levonen, J.J., Dillon, A.P. & Spiro, R.J. (Eds.) *Hypertext and Cognition. Hillsdale,* NJ: Lawrence Erlbaum Associates. [Online] Available: http://www-psych.nmsu.edu/~pfoltz/reprints/Ht-Cognition.html

**FRAKES, W.** (1992). Introduction to Information Storage and Retrieval Systems. In Frakes & Baeza-Yates, 1992a (pp. 1-12).

**FRAKES, W. & BAEZA-YATES, R.** (Eds.) (1992a). *Information Retrieval: Data Structures and Algortithms.* New Jersey: Simon & Schuster Company.

**FRAKES, W. & BAEZA-YATES, R.** (1992b). Preface. In Frakes & Baeza-Yates, 1992a (pp. vi-vii).

**FREI, H.-P.** (1996). *Information Retrieval — from Academic Research to Practical Applications.* [Online]. UBILAB, Union Bank of Switzerland. Available: http://www.ubs.com/

**GAMMA, E., HELM, R., JOHNSON, R. & VISSIDES, J.** (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

**GIARRATANO, J. & RILEY, G.** (1994). *Expert Systems: Principles and Programming (2nd ed.).* Boston: PWS Publishing Company.

**GLOOR, P.** (1997). *Elements of Hypermedia Design.* [Online] Available: http://www.birkhauser.com/hypermedia/

**GLOBAL REACH** (1999). *Global Internet Statistics (by Language).* [Online] Available: http://www.glreach.com/globstats

**GRAY, M.** (1997). *Internet Statistics: Web Growth, Internet Growth.* [Online] Available: http://www.mit.edu/people/mkgray/net/

**GREFENSTETTE, G.** (1994). *Explorationsin in Automatic Thesaurus Discovery.* Kluwer Academic Publishers.

**HARMAN, D., FOX, E., BAEZA-YATES, R. & LEE, W.** (1992). Inverted Files. In Frakes & Baeza-Yates, 1992a (pp. 28-43).

**HEARST, M.** (1994). Multi-Paragraph Segmentation of Expository Text. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics,* Los Cruces, NM, June. Available: ftp://parcftp.xerox.com/pub/hearst/acl94.ps.gz

**HEARST, M.** (1998). Automated Discovery of WordNet Relations. In *WordNet: An Electronic Lexical Database.* Christiane Fellbaum (Ed.). MIT Press.

**HEARST, M. & PLAUNT, C.** (1993). Subtopic Structuring for Full-Length Document Access. In *Proceedings of SIGIR'93*, Pittsburgh, PA.

**HEID, U., JAUSS, S., KRÜGER, K. & HOHMANN, A.** (1996). Term extraction with standard tools for coprus exploration. In *4th International Congress on Terminology and Knowledge Engineering*,Wien, August 26-30, 1996.

**HERSH, W., PRICE, P, KRAEMER, D., CHAN, B., SACHEREK, L. & OLSON, D.** (1999). A Large-Scale Comparison of Boolean vs. Natural Language Searching for the TREC-7 Interactive Track. [Online] In *Proceeding of the 7th Annual Text REtrieval Conference (TREC-7).* Available: http://trec.nist.gov/

**HPKB** (1999). About HPKB. [Online]. Available: http://projects.teknowledge.com/HPKB/about.html

**INTERNATIONAL ORGANISATION FOR STANDARDISATION** (1986). *Documentation - Guidelines for the Establishment and Development of Monolingual Thesauri (ISO 2788).* Author. Geneva.

**JAVA PLATFORM (A.K.A. JAVA 1.0/1.1).** [Computer software]. (1997). [Online] Sun Microsystems. Available: http://www.javasoft.com/

**JAVA 2 PLATFORM (A.K.A. JAVA 1.2).** [Computer software]. (1997). [Online] Sun Microsystems. Available: http://www.javasoft.com/

**KARP, P. D. & PALEY, S. M.** (1995). Knowledge Representation in the Large. [Online] In *Proceeding of the IJCAI.* Available: http://www.ai.sri.com/pubs/papers/Karp95-751:Knowledge/document.ps.Z

**KATZ, B.** (1997). *From Sentence Processing to Information Access on the World Wide Web.* [Online] Available: http://www.ai.mit.edu/people/boris/webaccess/

**KAVANAGH, J.** (1995). *The Text Analyzer: A Tool for Extracting Knowledge from Text.* MSc. Thesis, Dept. of Computer Science, University of Ottawa.

**KAVANAGH, J.** (1998). *The Text Analyzer.* Unpublished Internal Report, Dept. of Computer Science, University of Ottawa.

**KALGARRIFF, A.** (1997a). *Foreground and Background Lexicons and Word Sense Disambiguation for Information Extractions.* [Online] Available: cmp-lg/9712007.

**KALGARRIFF, A.** (1997b). *What is Word Sense Disambiguation Good For?* [Online] Available: cmp-lg/9712008.

**KENT, R. E. & NEUSS, C.** (1997). *Creating a WAVE (Web Analysis and Visualization Environment).* [Online] Available: http://wave.eecs.wsu.edu/

**KEYES, J. G.** (1999). *Using Conceptual Categories of Questions to Measure Differences in Retrieval Performance.* [Online] Available: http://ils.unc.edu/keyes/papers/asis.html

**KIRK, E.** (1997). *Information Retrieval and the Internet.* [Online] Available: http://milton.mse.jhu.edu:8001/research/education/retrieval.html

**LAKE RESEARCH GROUP** (1999). *Language Analysis and Knowledge Engineering Research Group (LAKE).* [Online] Available: http://dockman.site.uottawa.ca/

**LAGER, T.** (1995). *A Logical Approach To Computational Corpus Linguistics.* [Online] Sweden: Gothenburg Monographs In Linguistics 14. Available: http://www.ling.gu.se/~lager/taglog.html

**LAZARINIS, F.** [Computer software]. (1997). *Porter.java.* [Online] Address: Psilovraxou 12, Agrinio, 30100.

**LENAT, D. B. & GUHA, R. V.** (1989). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* New York: Addison-Wesley Publishing Company, Inc..

**LENAT, D. B.** (1995). Steps to Sharing Knowledge. In N.J.I. Mars (Ed.), *Towards Very Large Knowledge Bases.* IOS Press.

**LENAT, D. B, GUHA, R. V., PITTMAN, K., PRATT, D. & SHEPHERD, M.** (1990, August). CYC: Towards Programs with Common Sense. In *Communications of the ACM*, Vol. 33 No. 8. (pp. 30-50).

**LESK, M.** (1995). *The Seven Ages of Information Retrieval.* [Online] Available: http://ifla.org/VI/5/op/udtop5/udtop5.htm

**LINKSET LTD.** (1995). *Problems with Hypertext Usability?* [Online] Author. Available: http://www.linkset.co.uk/navig.htm

**LUGER, G. F. & STUBBLEFIELD, W. A.** (1993). *Artificial Intelligence:Structures and Strategies for Complex Problem Solving (2nd ed.).* California: The Benjamin/Cumming Publishing Company.

**MANBER, U. & WU, S.** (1994). GLIMPSE: A Tool to Search through Entire File Systems. [Online] In *Winter USENIX Technical Conference.* Available: ftp://ftp.cs.arizona.edu/glimpse/glimpse.ps.Z

**MEYER, I. & STEELE, J.** (1990). The Presentation of an Entry and Super-entry in an Explanatory Combinatorial Dictionary of English. In Steele, J. (Ed.), *Meaning-Text Theory: Linguistics, Lexicography, and Implications.* Ottawa: University of Ottawa Press.

**MCENERY, T. & WILSON, A.** (1996). *Corpus Linguistics.* Edinburgh: Edinburgh University Press.

**McEnery, T., Wilson, A. & Baker, P.** (1997). *Research Issues in Applied Linguistics.* [Online] Available: http://www.sal.tohoku.ac.jp/~gothit/corpus1/index.htm

**Mendelson, E.** (1997, April 8). Off-Line Search Utilities. *PC Magazine*, pp. 227-231.

**Microsoft Corporation** (1997). *The Microsoft Object Technology Strategy: Component Software.* Part No. 098-55163. Availalbe: http://www.microsoft.com/oledev/olemkt/oleent/obstrat2.html

**Minski, M.** (1975). "A framework for representing knowledge." In Winston, P. (ed.), *The Psychology of Computer Vision.* New York: MacGraw-Hill (pp. 211-280).

**Mollá, D.** (1999). *ExtrAns: An Answer Extraction System for Unix Manpages — On-line Manual.* [Online] Available: http://www.ifi.unizh.ch/CL/molla/secondy/

**NCITS.T2 Committee on Information Interchange and Interpretation** (1999a). *Knowledge Interchange Format.* [Online] Draft proposed American National Standard (dpANS) NCITS.T2/98-004. Author. Available: http://logic.stanford.edu/kif/dpans.html

**NCITS.T2 Committee on Information Interchange and Interpretation** (1999b). *Conceptual Graph Standard.* [Online] Draft proposed American National Standard (dpANS) NCITS.T2/98-003. Author. Available: http://www.bestweb.net/%7Esowa/cg/cgdpans.htm

**Neches, R., Fikes, R., Plum, T., Gruber, T., Patil, R., Senator, T. & Swartout, R.** (1991, fall). *Enabling Technoogy for Knowledge Sharing.* In AI Magazine.

**O'Brien, G. W.** (1994). *Information Management Tools for Updating an SVD-Encoded Indexing Scheme.* Technical Report CS-94-258. Msc. Thesis, Computer Science Department, Univeristy of Tenessee.

**Okurowski, M. E.** (1993, September). Information Extraction Overview. In *Proceedings of the TIPSTER Text Program (Phase I).* CISTI.

**Oracle** [Computer software] (1999). Oracle Corporation. Available: http://www.oracle.com/

**Parallel Understanding Systems Group** (1999a). *SHOE.* [Online] Department of Computer Science, University of Maryland at College Park. Available: http://www.cs.umd.edu/projects/plus/SHOE/

**Parallel Understanding Systems Group** (1999b). *Large-Scale Knowledge Representation:The PARKA Project.* [Online] Department of Computer Science, University of Maryland at College Park. Available: http://www.cs.umd.edu/projects/plus/Parka/

**Pilto, R.** (1990). *The Role of Structure in Text Accessibility—Contrasting Text and Hypertext.* [Online] Available: http://www.ekl.oulu.fi/staff/risto/HTACC/VER2/ARTICLE2.HTM

**PORTER, B., LESTER, J., MURRAY, K., PITTMAN, K., SOUTHER, A., ACKER, L. & JONES, T.** (1988). Research in the Context of a Multifunctional Knowledge Base: The Botany Knowledge Base Project. University of Texas at Austin.

**PRPIC, J.** (1997a). *DocKMan's Knowledge Base Builder.* MSc. Thesis, Dept. of Computer Science, University of Ottawa.

**PRPIC, J.** (1997b), *How to use DocKMan.* Unpublished Internal Report, Dept. of Computer Science, University of Ottawa, October 1997.

**QUILLIAN, M.** (1968).Semantic Memory. In Minsky, M. (ed.), *Semantic Information Processing*, MIT Press (pp 227-270).

**RKF** (2000). Proposer Information Package for BAA99-35: Rapid Knowledge Formation (RKF) Project [Online]. Available: http://projects.teknowledge.com/RKF/rkf_pip.html

**ROBIE, J., LAPP , J. & SCHACH, D.** (1998). *XML Query Language (XQL).* [Online] Available: http://www.w3.org/TandS/QL/QL98/pp/xql.html

**RUNDELL, M.** (1996). *The Corpus of the Future, and the Future of the Corpus.*[Online] Available: http://www.ruf.rice.edu/~barlow/futcrp.html

**RUTKOWSKI, M. L. & STASKO, S. R.** (1999). *Data, Information, Knowledge — Understanding the Difference.* [Online] The Knowledge Chronicle. Available: http://www.walshcol.edu/~mrutkow/article1A.htm

**SALTON, G., ALLAN, J. & BUCKLEY, C.** (1993). *Approaches to Passage Retrieval in Full Text Information Systems.* [Online] Available: http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR93-1334

**SALTON, G. & MCGILL, M.** (1983). *Introduction to Modern Information Retrieval.* New York: McGraw-Hill Book Company.

**SALTON, G.** (1988). *On the Use of Spreading Activation Methods in Automatic Information Retrieval.* Technical Report 88-907. Department of Computer Science, Cornell University.

**SALTON, G.** (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* California: Addison-Wesley Publishing Company.

**SANDERSON, M. & CROFT, B.** (1998). *Deriving Concept Hierarchies from Text.* [Online] Available: http://cobar.cs.umass.edu/pubfiles/ir-156.ps

**SENG, T. L.** (1999). *Knowledge Representation of Rule-Based Expert Systems.* [Online] Available: http://www.comp.nus.edu.sg/~johnm/ic52a5/as/l20_limteese.htm

**SKUCE, D.** (1992). *Notes on ClearTalk.* AI Laboratory, Dept. of Computer Science, University of Ottawa.

**SKUCE, D.** (1995a, September). Knowledge Management in Software Desing: A Tool and a Trial. In *Software Engineering Journal* (pp. 183-192).

**SKUCE, D.** (1995b). Conventions for Reaching Agreement on Shared Ontologies. In *Proceeding of KAW.* Banff.

**SKUCE, D.** (1997). *Integrating Web-Based Documents, Shared Knowledge Bases, and Local Document Searching for User Help.* Dept. of Computer Science, University of Ottawa.

**SKUCE, D. & LETHBRIDGE, T. C.** (1997). *CODE4: A Unified System for Managing Conceptual Knowledge.* AI Laboratory, Dept. of Computer Science, University of Ottawa.

**SKUCE, D.** (1998). DocKMan's Knowledge Base Builder. In *Proceeding of KAW98*, April 1998. Banff.

**SKUCE, D.** (1999a). Toward Unified Knowledge Management: Integrating Documents, Knowledge Bases, Databases, and Linguistic Knowledge. In *Eleventh Workshop on Knowledge Acquisition, Modelling and Management.*

**SKUCE, D.** (1999b). *ClearTalk Conventions.* Internal Report, Dept. of Computer Science, University of Ottawa.

**SMADJA, F.** (1993). Retrieving Collocations from Text: Xtract. *Computational Linguistics, Vol. 19, No. 1.*

**SMEATON, F. & AGOSTI, M.** (Eds.). (1994). *Information Retrieval and Hypertext.* Kluwer Academic Publishers.

**SNEIDERS, E.** (1998, July). Information Systems in the WWW Environment. [Online] IFIP TC8/WG8.1 Working Conference, 15-17 (pp.298-319). China: Chapman & Hall. Available: http://ekd.dsv.su.se/ifip98/faq-answering.htm

**SOWA, J. F.** (1995). *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Unpublished Draft.

**SRI INTERNATIONAL** (1997). *Open Knowledge Base Connectivity Home Page.* [Online]. Available: http://www.ai.sri.com/~okbc/

**STEINBERG, S.** (1997). Seek and Ye Shall Find (Maybe). [Online] *HotWired* 4.05.

**SULLIVAN, D.** (1999). *Search Engine Sizes.* [Online]. Search Engine Watch. Available: http://www.searchenginewatch.com/reports/sizes.html

**TOWELL, G. VOORHEES, E. M., GUPTA, N. K. & JOHNSON-LAIRD, B.** (1997). *Slearning Collection Fusion Strategies for Information Retrieval.* Siemens Corporate Research, Princeton, NJ.

**VAN RIJSBERGEN, C. J.** (1979). *Information Retrieval* (2nd Ed.). [Online]. *(Original site no longer available).*

**VAN RIJSBERGEN, C. J.** (1986). A new theoretical framwork for information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* 23-29.

**VOORHEES, E. M. & HARMAN, D.** (1998). Overview of the Seventh Text REtrieval Conference (TREC-7). [Online] In *Proceeding of the 7th Annual Text REtrieval Conference (TREC-7).* Available: http://trec.nist.gov/

**VOUTILAINEN, A. & SILVONEN, M.** (1996). *A Short Introduction to ENGCG.* [Online] Available: http://www.lingsoft.fi/doc/engcg/into/

**YANG, K.** (1997). *Combining Multiple Document Representations and Multiple Relevance Feedback Methods to Improve Retrieval Performance.* [Online] MSc. Thesis, School of Information and Library Science,University of North Carolina. Available: http://ils.unc.edu/yangk/thesis/thesis.htm

**YANG, K., MAGLAUGHLIN, K., MEHO, L., & SUMNER, R. G. JR.** (1998). IRIS at TREC-7. [Online] In *Proceeding of the 7th Annual Text REtrieval Conference (TREC-7).* Available: http://trec.nist.gov/

**WAGNER, G.** (1999). *Foundations of Knowledge Systems with Applications to Databases and Agents.* [Online]. Boston: Kluwer Academic Publishers. Available: http://www.infinformatik.uni-leipzig.de/~gwagner/ http://www.inf.fu-berlin.de/~wagnerg/

**WHITEHEAD, S. D.** (1995, December). Auto-FAQ: an Experiment in Cyberspace Leveraging. In *Computer Networks and ISDN Systems*, Vol.28, no.1-2 (pp.137-146).

**WITTEN, I., MOFFAT, A. & BELL, T. C.** (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images.* San Francisco: Morgan Kaufmann Publishing.

**WOODS, W. A.** (1997). *Conceptual Indexing:A Better Way to Organize Knowledge.* [Online] The SML Technical Report Series Technical Report TR-97-61. USA: Sun Microsystems Laboratories.

**WORDSMITH TOOLS.** [Computer software]. (1999). [Online] Available: http://www.liv.ac.uk/~ms2928/homepage.html

**WORLD WIDE WEB CONSORTIUM** (1992). *The Project.* [Online] Author. Available: http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html

**WORLD WIDE WEB CONSORTIUM** (1999). *About The World Wide Web.* [Online] Author. Available: http://www.w3.org/WWW/

**ZAYOUR, I.** (1997). *Information Retrieval over the World Wide Web.* MSc. Thesis, Dept. of Computer Science, University of Ottawa.