
[All ETDs from UAB](#)

[UAB Theses & Dissertations](#)

1998

Automated knowledge acquisition of case-based semantic networks for interactive enhancement of the data mining process.

Aubrey Eugene Hill
University of Alabama at Birmingham

Follow this and additional works at: <https://digitalcommons.library.uab.edu/etd-collection>

Recommended Citation

Hill, Aubrey Eugene, "Automated knowledge acquisition of case-based semantic networks for interactive enhancement of the data mining process." (1998). *All ETDs from UAB*. 6209.
<https://digitalcommons.library.uab.edu/etd-collection/6209>

This content has been accepted for inclusion by an authorized administrator of the UAB Digital Commons, and is provided as a free open access item. All inquiries regarding this item or the UAB Digital Commons should be directed to the [UAB Libraries Office of Scholarly Communication](#).

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

AUTOMATED KNOWLEDGE ACQUISITION OF CASE-BASED SEMANTIC
NETWORKS FOR INTERACTIVE ENHANCEMENT OF THE DATA MINING
PROCESS

by

AUBREY E. HILL

A DISSERTATION

Submitted to the graduate faculty of the University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

1998

UMI Number: 9839837

**Copyright 1998 by
Hill, Aubrey Eugene**

All rights reserved.

**UMI Microform 9839837
Copyright 1998, by UMI Company. All rights reserved.**

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

Copyright by
Aubrey E. Hill
1998

ABSTRACT OF DISSERTATION
GRADUATE SCHOOL, UNIVERSITY OF ALABAMA AT BIRMINGHAM

Degree Ph.D. Program Computer and Information Sciences

Name of Candidate Aubrey E. Hill

Committee Chair Warren Jones

Title Automated Knowledge Acquisition of Case-Based Semantic Networks for
Interactive Enhancement of the Data Mining Process

This work presents a knowledge acquisition algorithm for the automated construction of a case-based semantic network model from conventional relational databases. An associative retrieval algorithm is also provided to support interactive use of the model. Scalability of the performance of the knowledge acquisition algorithm is investigated with a parallel version. In addition to acquiring explicit information, which is represented as cases, the knowledge acquisition algorithm also captures the semantic and associative relationships which are implicit in the database. The semantic links define set membership for the acquired cases and the associative relationships define concepts. A user friendly graphical user interface has been developed to facilitate access to the knowledge base. A browsing facility, which supports navigation of semantic links, is particularly suited for enhancing interactivity of the data mining process with original data after interesting patterns have been identified.

ACKNOWLEDGMENTS

I would like to acknowledge and thank those who have helped make this research possible. First, I thank my advisor, Dr. Warren Jones, for his flexibility and for bringing me into the research group in the summer of 1996. Also, I would like to thank the other members of my committee: Dr. Sanjay Singh, for his encouragement; Dr. Mike Hardin, for his advice and discussions of Bayesian techniques; Dr. Barrett Bryant, for his editing and comments; and Dr. Robert Hyatt, for his advice on parallelism.

I would also like to acknowledge and thank Dan Austin of BellSouth, for his flexibility and friendship; Lewis Hughes of BellSouth, for the use of his multiprocessor computer; my wife, Donna, for her support during the writing of this dissertation; and, finally, God, for providing the means at every step of the way towards this degree.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER	
1 INTRODUCTION	1
Purpose.....	1
The Need for Knowledge.....	1
Declarative Knowledge Versus Procedural Knowledge.....	3
The Problem of Brittleness	5
Objectives	7
2 PREVIOUS AND RELATED WORK.....	9
CBR	9
Origins of CBR	10
Case-Based Learning	16
Knowledge Discovery/Data Mining and Machine Learning	17
3 THE KNOWLEDGE BASE ARCHITECTURE.....	21
The Conceptual Knowledge Base.....	24
The Database Implementation of the Knowledge Base.....	27
4 KNOWLEDGE ACQUISITION	33
A More Detailed Description of the Algorithm	37
Counting duplicate input records.....	37
Determining if a case already exists	37
Determining if an item already exists in the IS_A_Table.....	38
Dynamic recalculation of the links	39

TABLE OF CONTENTS (Continued)

	<u>Page</u>
CHAPTER	
An Example to Demonstrate the Knowledge Acquisition Algorithm	40
A Parallel Version of the Knowledge Acquisition Algorithm.....	44
5 PERFORMANCE EXPERIMENTS	47
6 THE ASSOCIATIVE RETRIEVAL PROCESS	55
7 CONCLUSIONS AND FUTURE WORK	67
LIST OF REFERENCES	72

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1 Example of Input Data	25
2 The Case Base Table	28
3 IS_A_TABLE.....	28
4 CASE_IS_A_TABLE	29
5 DB_ASSOCIATIONS_TABLE.....	29
6 Input Records	41
7 The CASE_BASE Table	41
8 The CASE_IS_A Table	42
9 DB_ASSOCIATION Table.....	43
10 IS_A Table	44

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1 Nearest neighbor similarity metric	14
2 Conceptual view of the knowledge base	26
3 A high level view of the algorithm.....	34
4 Comparison of serial and parallel versions for 4 attributes.....	46
5 A high level view of the parallel algorithm.....	48
6 Comparison of one-process versus two-processes as a function of the number of attributes (time in seconds)	48
7 Comparison of serial and parallel versions for one attribute.....	49
8 Comparison of serial and parallel versions for two attributes.....	49
9 Comparison of serial and parallel versions for three attributes.....	50
10 Comparison of serial and parallel versions for five attributes.....	50
11 Performance of one process as a function of number of attributes	51
12 Performance of two processes as a function of number of attributes.....	51
13 Effect of database reorganization on performance.....	53
14 Performance of the strength calculation routines	54
15 The initial screen	56
16 The initial search window	59
17 The case details window.....	61
18 Examples of IS_A_Organism.....	62
19 Following relationship links.....	64

LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Page</u>
20 CASE_STACK implementation	65
21 Example of minimized detail display windows	66

CHAPTER 1

INTRODUCTION

Purpose

This research proposes an approach to the problem of constructing artificially intelligent systems with the common world knowledge which they require to become increasingly intelligent and useful. It describes an algorithm for the automated acquisition of declarative knowledge from existing databases. An architecture for the resulting memory is also presented. The algorithm, in conjunction with this memory architecture, will allow the system to add continually to the contents of its cumulative, persistent, probabilistically associative, case-based semantic memory. In addition to the knowledge acquisition algorithm and the memory architecture, an algorithm for the retrieval of items from memory is also presented. This retrieval mechanism is designed to work with the memory to provide probabilistic recall and intelligent traversal of the entire memory.

The Need for Knowledge

Knowledge-based systems are based on the physical symbol system hypothesis (Laird, Newell, & Rosenbloom, 1987; Newell & Simon, 1976). This hypothesis states that a general intelligence must be realized with a symbolic system. In this viewpoint the physical world is represented as a mental model consisting of a collection of symbols and their interrelationships. The knowledge-based system “reasons and thinks” by manipulating its internal mental model. This manipulation often consists of simulation

and/or search by the activation and following of causal or associative links in the mental model. This hypothesis is related to the characterization of computer programs as consisting of data structures and algorithms. Computer implementations of such mental models include Patil's (1981) causal model of electrolyte and acid-base disorders and Hill's (1987) AMEX system for the diagnosis of malfunctions in an ammonia synthesis plant. Ford, Bradshaw, Adams-Webber, and Agnew (1993) have made the case that knowledge acquisition is actually an exercise in domain modeling.

Knowledge-based systems often perform tasks such as diagnosis, which in humans requires the possession of a body of knowledge and experience referred to as expertise. It has been acknowledged that artificial intelligence (AI) depends upon the possession of large amounts of knowledge, both domain specific and general world knowledge. "Artificial intelligence" and "knowledge based" are often used synonymously. Minsky and Papert (1974), as quoted in Harris (1985, p. 9), contrast the knowledge based approach to AI to the previously favored power-based approach as follows:

The power strategy seeks a generalized increase in computational power. It may look toward new kinds of computers. . . The knowledge strategy sees progress as coming from better ways to express, recognize, and use diverse and particular forms of knowledge. . . . The view that the process of intelligence is determined by the knowledge held by the subject.

Paul Cohen and Edward Feigenbaum (1981, Preface xv) note in the preface to volume 3 of their Handbook of Artificial Intelligence, that "The power of an artificial intelligence program is directly proportional to what it knows. . . . The performance of learning programs is directly proportional to what they know." Riesbeck (1979, p. 409) notes the need for knowledge in natural language processing: "Understanding natural language texts requires knowledge--lots of it. It requires . . . knowledge about the physical world." Harris (1985) makes the case that world

knowledge is needed for pragmatic analysis (in addition to the usual syntactical and semantic analysis) in natural language processing. She defines pragmatics as involving knowledge of the physical world, frames of reference, and the context in which a single statement is made. The importance of knowledge for the realization of AI is the same as she defines for natural language understanding, which is that a great deal of the information that people use is not stated explicitly but can be inferred from what is stated. As an example, if it was stated that someone had fallen in the water, most people could infer wetness as a result. This is possible because people have stored the fact that water has the property of imparting wetness to almost any object that it touches. In contrast, a knowledge-lean computer program might simply assert the fact that the person had changed position from out of water to in water. Benjamin Kuipers (1979), one of the leading researchers in the field of qualitative simulation, defines commonsense knowledge as

knowledge about the structure of the external world that is acquired and applied without concentrated effort by any normal human that allows him or her to meet the everyday demands of the physical, spatial, temporal and social environment with a reasonable degree of success. (p. 394)

Declarative Knowledge Versus Procedural Knowledge

The knowledge used by AI programs is typically classified as being either declarative or procedural. Declarative knowledge is the static knowledge about events, objects, relationships, and the physical world. Procedural knowledge is “know how” or “how to” knowledge. Goldstein and Papert (1977) describe procedural knowledge as follows: “An item of knowledge, a concept . . . is seen as an active agent rather than as a passive manipulatable object.” (p. 98).

As a simple example, a subroutine which adds two numbers is procedural knowledge, whereas a table-lookup of sums uses the declarative knowledge stored in the table. Declarative knowledge is usually explicit, while procedural knowledge is implicit. Both types of knowledge are needed. Declarative knowledge is usually thought of as the data structures of a program, whereas procedural knowledge is thought of as the code that manipulates those structures to reach new conclusions. Declarative knowledge without procedural knowledge is passive and does not lead to new inferences. Totally procedural systems often resemble single purpose black boxes that cannot explain their reasoning or describe their knowledge. They are not as likely to be able to “muse” (as described by Fertig and Gerlenter, 1988, who coined the term “musing databases”) about what they know. Bjork and Bjork (1996, p. 364), in their book, Memory, stress the importance of declarative knowledge: “A major component of thinking seems to be the possession of accessible and usable declarative knowledge.”

Preserving knowledge in an explicit, declarative form, as opposed to the derivation of rules or decision trees, is an issue of representation. While representation may not matter to the computer, it is very important for the interaction with humans. It is easy to see that in many instances a particular system could be implemented as a neural network, a set of production rules, a case base, or a semantic network. However, case bases and semantic networks are much more understandable than rules or neural networks. Case bases and semantic networks also lend themselves to the production of explanations. Representation is a very important issue. Findler (1979) wrote that the structure of knowledge representation is of paramount importance and that the success of the project concerned critically depends on it.

The Problem of Brittleness

In 1997 the chess-playing program, Deep Blue, defeated the reigning human chess champion, Gary Kasparov. Following this event, Kasparov was asked for his thoughts and analysis of the match, which he provided. By the standard of chess-playing ability, Deep Blue is more intelligent than Kasparov; however, no one considered asking Deep Blue for its thoughts and analysis. This example points out one of the major problems with AI programs and especially with expert systems--brittleness. Brittleness is defined as abrupt failure when a system reaches the limits of its domain knowledge. Therefore, while Deep Blue possessed the procedural knowledge to be extremely competent at chess, it did not have the declarative world knowledge and natural language processing capability to be able to comment on its recent encounter with Kasparov. The need for world knowledge is being addressed in a number of ways. For expert systems in limited domains, human experts are interviewed to obtain expertise. This is usually an inefficient, expensive, and time-consuming approach to knowledge acquisition. Domain experts are valuable, and their available time is limited and fragmented. Often, those with the most expertise are the least introspective. Among the other approaches are the CYC project (Lenat & Guha, 1990), which is attempting to hand code enough information so that the system can then begin to acquire information on its own. The SNOWY project (Gomez, Hull, & Segami, 1994) attempts knowledge acquisition by reading encyclopedia articles. A number of techniques in the categories of machine learning, data mining, and knowledge discovery address the knowledge acquisition problem. Among these are the creation of decision trees, rules by induction from a large number of examples, and the construction of Bayesian belief networks. Langley and Simon (1995) list the following

five basic learning paradigms: (a) neural networks, (b) case-based reasoning (CBR), (c) genetic algorithms, (d) rule induction, and (e) analytic learning.

While each of these has produced impressive results, the end result, with the possible exception of CBR, is not what is usually intended when it is said that a human has learned something. Neural nets, genetic algorithms, rule induction, and analytic learning all result in rules, functions, or optimal parameters. CBR results in the clustering of episodic memories. The intended use of most of these techniques is the creation of systems which can classify new inputs based on past training examples. In practice, most of the learning is procedural, limited to a single domain, and not cumulative. To begin the process of building intelligent systems will require that these systems be capable of acquiring both procedural and declarative knowledge in multiple domains. This knowledge acquisition should be cumulative so that the system becomes increasingly knowledgeable.

The difficulty of knowledge acquisition has resulted in the descriptive term, “knowledge acquisition bottleneck.” Entire books (Eysenck & Keane, 1995) have been devoted to the techniques for acquiring knowledge.

Ideally, artificially intelligent systems would learn nouns and fundamental concepts by direct observation of the physical world, in conjunction with labeling by a “parent” or “teacher.” This fundamental knowledge could then be used to bootstrap the learning-by-reading process. This learning by direct observation is not currently possible. This research proposes the learning of nouns and concepts by a process of automated knowledge acquisition from conventional databases.

Objectives

In general, the proposed techniques could result in populated memories which could serve as a basis/testbed for various AI projects. A generic associative memory with known mechanisms for knowledge acquisition, storage, organization, and recall could serve as a starting point for subsequent AI research. A more specific, practical use would be as a knowledge acquisition tool for knowledge-based expert systems.

At a conceptual level the memory will be represented as a dynamic, probabilistically associative, case-based semantic network. At a lower level it will utilize a conventional database management system. The concurrent implementation of these three representations—(a) probabilistic associations, (b) cases as the elemental item in the memory, and (c) semantic/associative networks--will allow several modes of reasoning. Among these are (a) simple associative recall, (b) CBR/recall, (c) model-based reasoning/qualitative simulation, and (d) the use of hierarchies/taxonomies. The implementation of cases will simulate human episodic memory (Eysenck & Keane, 1995), while the construction (where possible) of named links such as “is-a” and “causes” will simulate human semantic memory (Quinlan, 1966).

This work has the following specific objectives: (a) the design of an algorithm for the conversion of conventional databases to declarative knowledge bases, (b) the design of a parallel version of this database conversion algorithm, (c) the comparison of the performance of serial and parallel versions of the algorithm, (d) a conceptual model and implementation model for the resulting probabilistically associative case-based semantic memory, (e) an algorithm to retrieve items from memory which utilizes the probabilistic nature of the knowledge base, (f) a demonstration of the knowledge base as an intelligent

database, and (g) proposing an enhanced interactivity version of the data mining process model.

In chapter 2, we discuss related research. Chapter 3 describes the knowledge base architecture. The knowledge acquisition algorithm is presented in chapter 4, and a parallel version is discussed in chapter 5. Chapter 6 describes the associative retrieval process. Finally, conclusions and future work are presented in chapter 7.

CHAPTER 2

PREVIOUS AND RELATED WORK

This work is positioned at the intersection of a number of subdisciplines of computer science. It was originally conceived from the perspectives of CBR and data mining. It specifically addresses the processes of case acquisition and case retrieval. Case-based retrieval, in turn, is related to the area of associative memory. The problems of inserting new items into an associative memory and retrieving these same items suggested both the memory architecture and its retrieval mechanisms. There is also intersection with the disciplines of knowledge-based systems, machine learning, and intelligent databases.

CBR

CBR can be viewed as a knowledge representation technique, a reasoning mechanism, and a machine learning technique. Leake (1996) describes CBR as reasoning based on remembering versus reasoning by the chaining together of generalized rules. He describes the knowledge source for CBR as a collection of stored cases recording specific prior episodes. Kolodner (1993) describes CBR as a reasoning strategy in which the reasoner remembers previous situations similar to the current one and uses them to help solve the new problem. According to Leake (1996), CBR is based on two principles: (a) The world is regular and similar problems have similar solutions, and (b) the problems tend to recur. The full cycle of CBR (Kolodner, 1993) involves the following steps:

1. Retrieve cases from the collection of cases.
2. Propose an approximate solution.
3. Adapt the approximate solution to fit the new situation.
4. Criticize the retrieved solution to determine if it is the best alternative.
5. Evaluate the goodness of the solution in the real world.

Cases are the basic unit of knowledge in CBR. A case is a thing or a situation which is remembered. Kolodner and Leake (1996) define a case as a contextualized piece of knowledge representing an experience that teaches a lesson. A collection of cases forms a case base which serves as an episodic memory of specific events. The retention of specific memories as opposed to generalizations distinguishes episodic memory from semantic memory. For example, the memory of a specific person is episodic, whereas the memory of the general characteristics of people is an example of semantic memory. Individual cases can be represented as objects or records, or as entries in a database. A very useful definition of a case base is a database which allows fuzzy or ambiguous queries (CBR Express for Windows User's Guide, 1990-1995). Case-based methods are used in legal, medical and business education and in the practice of these professions. In the legal field much of the decision-making process is based on discovering precedents. Medical diagnosis often consists of matching symptoms to diseases and retrieving a diagnosis from memory. Many business schools use actual cases as a teaching method.

Origins of CBR

A number of researchers have proposed structures which could be called CBR. The most obvious path to the development of CBR as it is currently defined begins with the work of Roger Schank and Robert Abelson (1977). Schank's student, Janet

Kolodner, further refined the concept and established it as a separate subdiscipline of computer science. Schank, a linguist by training, in the course of developing programs which could understand natural language, proposed the idea of conceptual dependency (Schank, 1973). In conceptual dependency theory, concepts are used to guide natural language understanding by generating conceptual expectations. As an analogy to the parser component of a compiler, which has certain language-dependent syntactical expectations based on previous input, a natural language processing system has certain conceptual expectations based on what concepts have occurred previously. A compiler, having seen a left parenthesis, would expect to encounter a matching right parenthesis. A natural language system, based on conceptual dependency theory, having seen the sentence, "The ball was thrown across the plate," would use its conceptual knowledge to determine first that the plate in question was a piece of baseball equipment and not something used to hold food. Having this piece of information would, in turn, help determine that the ball was probably a baseball. Purely syntactical analysis could never disambiguate the two senses of "plate." Schank and Abelson continued to elaborate on their contention that true natural language understanding required episodic world knowledge (Schank & Abelson, 1977; Schank, 1982). They proposed that the understanding of stories involved the use of scripts, plans, and goals to produce expectations which were then checked for conceptual consistency with the previous input. In their 1977 book, Schank and Abelson describe a script as a standardized, generalized episode. In the 1982 book, Dynamic Memory (Schank, 1982), their refined, practical definition of a script was, "a data structure that was a useful source of predictions." Plans are described as general information about how actors achieve goals. Schank and Abelson (1977) introduce goals to

generate expectations of likely events from contextual information about the characters and from well developed belief systems about the world.

Their idea of a dynamic memory is based on the concept that we use what we know to process new information. New input matches some existing structure in memory, causes a modification to that structure, or causes a new structure to be created. This is referred to as failure-driven memory. Schank and Abelson's (1997) view of such a memory is not that of a passive repository from which facts are retrieved but rather that of a dynamic organization in which reminding, understanding, and learning are all closely related processes. Reminding is finding the correct memory structure to process an input. Understanding is defined as finding the closest match to past experience for an input, and coding that input in terms of the previous memory, indicating the difference between the new input and the matching memory. Any expectation failure results in learning, where learning is defined as a modification of memory structures. If none of the existing memory structures provides expectations which are met by the current input, then the current input must be something that has not been previously encountered; therefore, it is deemed to be interesting. Interesting items are remembered.

Schank (1982) elaborates on the types of memory structures which are necessary for understanding and learning. Specific memories are stored as scenes which consist of physical aspects and goals. Scenes are connected together by structures which he calls memory organization packets (MOPs). Scripts are attached to scenes to fill in the particulars of that scene. As MOPS organize scenes, so scenes organize scripts. Scenes provide general information about their attached scripts. The attached scripts provide the particular details for scenes. Thematic organization points (TOPs) are structures that represent abstract, domain-independent information. The ability to create TOPs is the key to

reminding, memory organization, and generalization. TOPs, being defined in terms of goals, plans, and themes, provide for cross-contextual reminders. All of these ideas concerning dynamic memory and the use of existing memory structures to process new inputs depend upon the successful retrieval of the appropriate items from memory. A key issue in successful retrieval is the assignment of the proper indexes to items in memory.

Beginning with these ideas Kolodner (1983) developed CYRUS, a case-based cognitive model of former Secretary of State, Cyrus Vance. Both Schank (1982) and Kolodner (1983) address the issue of indexing in an episodic memory. Schank (1982) describes indexing as the ultimate problem of memory. Indexes, also referred to as the labels of a case (Kolodner, 1996), are their important or defining sets of features. Indexes are those sets of features which distinguish individual cases from others. As Kolodner (1996) points out, it is important to understand the difference between the term index, as used in the context of CBR, and the term index as used in computer science. As stated, in the context of CBR an index is a distinguishing characteristic or a feature which makes an case memorable. In computer science and database technology the term index is usually interpreted as "pointer," which means a memory address calculated from the content of the item to be stored. CBR uses indexes to find relevant cases. Database management and file systems use indexes to provide random access to records and to avoid exhaustive sequential searches. Of course, a distinguishing feature of a case can be indexed by a database management system to aid in finding particular records, but this is a separate issue from the relevance of that feature to case retrieval. Cases can be simultaneously indexed by various features. This can yield retrieval from various view-points.

Having collected a number of cases of interest and indexed them by their relevant and distinguishing features, a CBR system is faced with its primary task of retrieving

cases which are similar to the current input. This involves the process of matching and ranking. These processes require the use of a similarity metric for the comparison of cases. A simple nearest neighbor similarity metric is shown in Figure 1.

$$\frac{\sum_{i=1}^n w_i \times \text{sim}(f_i^I, f_i^R)}{\sum_{i=1}^n w_i}$$

Figure 1. Nearest neighbor similarity metric. In this metric, i represents each of the individual features, w is the importance of each feature, and sim is the similarity measurement for each feature. f^I is the value of the feature for the input and f^R is the value of the feature for the retrieved case. This calculation provides a means of comparing and ranking each retrieved case (from the total casebase) to the input.

Although the path from Schank and Abelson (1977) through Kolodner (1983) is one of the clearest routes to CBR, others have proposed similar approaches, although they have not always referred to their work as CBR. Among these is the MBRtalk project, which dealt with the pronunciation of English text (Stanfill & Waltz, 1986). Their premise is that the intensive use of memory to recall specific episodes from the past, rather than rules, should be the foundation of machine reasoning. The memory-based reasoning hypothesis is that reasoning may be accomplished by searching a database of worked problems for the best match to the problem at hand. They contrast their work to rule induction, which infers rules that reflect regularities in the data, by noting that their approach works directly from the database of examples rather than first deriving rules. They claim that memory-based reasoning differs from CBR and explanation-based reasoning in that their

approach does not require the use of a strong domain model. Having made this claim, they go on to

discuss the importance of context in the weighting of features and the interaction of features. Much of the emphasis by Stanfill and Waltz (1986) is on the use of parallelism in retrieval. At the time both were involved with The Connection Machine project. Their work seems to be CBR from a different viewpoint.

In his 1994 review article Aamodt (1994) notes that CBR is just one of a set of terms used to refer to similar types of systems. He lists several related types of reasoning which used past experiences as their basis:

1. Exemplar-based reasoning--which defines concepts extensionally as a set of its exemplars. The Protos system is an example (Porter & Bareiss, 1986).
2. Instance-based reasoning--a specialization of exemplar-based reasoning to a syntactic CBR approach. A large number of instances are needed to define a concept to compensate for the lack of general background knowledge. Simple representations such as feature vectors are used. The focus is on automated learning.
3. Memory-based reasoning--a large memory of many cases and parallel search are the emphasis of this approach. Often, storage and access are more syntactic than semantic.
4. Analogy-based reasoning--often used to solve problems by the use of cross-domain analogies versus the single domain emphasis of typical CBR.
5. CBR--typical CBR is distinguished from the previously mentioned methods by the richness of information contained in the cases, the background knowledge which is available, and the ability to modify and adapt retrieved solutions. This "full" CBR also has a basis in cognitive psychology.

Case-Based Learning

According to Kolodner (1993), case-based learning (CBL) is accomplished primarily by accumulating new experiences in memory and indexing them appropriately. Learning also takes place when failures are explained and repaired, when successful conclusions are explained, and when “good” indexes are generated. She holds the same view proposed by early work of Schank (1982) that learning is most often the result of the failure of expectations. It is important to distinguish the failure of expectations from the failure to achieve goals. In comparing the current input to a stored case(s), a reasoner will have generated certain conceptual expectations. If these expectations are not met, an expectation failure has occurred and must be explained and corrected. Learning has happened by the modification or addition to memory. To begin reasoning a minimal number of cases must be used to seed the case base. In many systems additional cases are learned when expectation failure occurs.

CBL algorithms are often called “lazy learning” algorithms because they store the entire training set and postpone all inductive generalization until classification time (Wettschereck, Aha, & Mohri, 1997).

Aha (1991) proposes a number of CBL algorithms and discusses the most common criticisms of this type of machine learning. In his view CBL algorithms are examples of supervised learning algorithms which output a concept description. They process a set of training cases and use them to classify new inputs into categories. The purpose of Aha's (1991) work was to correct the supposed deficiencies of CBL algorithms which had previously been outlined by the critics of CBL. The simplest type, CBL1, stores all cases after its preprocessor has normalized all numeric feature values. CBL2 differs from CBL1 in that it retains only incorrectly classified cases in its concept descriptions. This is done in

an attempt to reduce the computational load of computing similarity metrics for a large number of cases and to reduce the storage requirements. Aha's (1991) CBL3 algorithm determines which stored cases correctly classified new input. Only those cases which were successful were allowed to participate. The purpose of this modification is to gain a tolerance for noisy cases. In part to address the criticism that CBL algorithms are sensitive to the choice of similarity function, Aha's (1991) CBL4 learns a separate set of feature weight settings for each feature.

CBL algorithms allow incremental learning. In contrast to approaches such as artificial neural networks (ANNs), CBL allows the addition of new cases at any time without have to recalculate the weights of links between the nodes, as is done in ANNs. This allows a CBL to do cumulative learning. Of all the machine learning techniques, CBL is the closest to allowing the learning and use of declarative knowledge. Other techniques such as the induction of decision trees and ANNs tend to be inexplicable black boxes which cannot display their reasoning process or the original data from which it was derived. However, like most other machine learning techniques, the ultimate purpose of current efforts in CBL is to produce a system which can classify its inputs. While this is useful, the term machine learning generates expectations of more. To the uninitiated, machine learning implies having computers learn as humans learn. This should consist, at least in part, of learning declarative knowledge which can later be used for a number of purposes, including discussion, browsing, and musing.

Knowledge Discovery/Data Mining and Machine Learning

Knowledge discovery in databases and data mining (KDD) is defined as the overall process of discovering useful knowledge in data (Fayyad, Piatetsky-Shapiro, & Smith,

1996) and also as “the nontrivial extraction of implicit, previously unknown, and potentially useful information from data” (Frawley Piatetsky-Shapiro, & Matheus, 1991). In recent years KDD has been recognized as a new and useful area of research. The first KDD workshop was held in 1989. KDD has developed in response to the existence and constant growth of databases. The value of data is dependent upon being able to extract useful information for decision support or understanding the phenomenon from which the data were derived. Data mining is one step in the process (Fayyad et al., 1996) of knowledge discovery. The overall process has been described in terms of the following steps (John, 1997):

1. Understand the problem.
2. Extract the data
3. Clean/engineer the data
4. Engineer a data mining algorithm.
5. Search for potentially interesting patterns by running the data mining algorithm.
6. Evaluate the patterns.

The data mining step makes use of specific algorithms for extracting patterns or models from data. The data mining algorithms are often borrowed from the field of machine learning.

Prediction and description are the two generally accepted high level goals of data mining according to Fayyad et al. (1996). Prediction/Description can be facilitated by constructing a model of the data and then using that model for prediction or description of the underlying data (Epstein, 1997). This is much like doing a simple linear regression

model of a number of data points. The resulting slope and intercept give both a description of the data points and can be used to predict y-values for given x-values.

According to this view (Epstein, 1997; Fayyad et al., 1996), there are a number of functions performed by the models resulting from data mining. These are (a) classification--the determination of what attributes/values constitute a specified category and the use of this knowledge to properly classify new inputs; (b) regression--building a model and then using that model to predict continuous values; (c) time-series forecasting--similar to regression, uses past data to attempt to predict future trends; (d) clustering--the grouping of members into groups or clusters which are not known when the process is started (clusters will be determined dynamically); (e) associations/link analysis--determining which things consistently occur together; (f) sequence discovery/analysis--finding associations over time; (g) summarization--seeks a concise description of the data; and (h) dependency modeling--searches for dependencies among variables. The models which result from data mining can be represented in a number of ways. Among the more commonly used representations are decision trees; classification rules; linear models; nonlinear models such as neural networks, example-based methods (CBR), probabilistic graphical dependency models (Bayesian networks); and relational attribute models (Wu, 1995). Almost all of these techniques are intended either to classify new inputs or to predict future values. No research has been identified which proposes to provide an interactive facility to support browsing a declarative knowledge base of the original data within the context of interesting patterns identified by data mining.

This research also overlaps with the construction of ontologies. The dictionary defines ontology as the branch of philosophy that deals with being. Ontological engineering (building knowledge bases) is the fundamental problem being addressed by the CYC

project (Lenat & Guha, 1990). Copeland (1997) gives a review of the progress of CYC and its prospects. The current problem with CYC and other ontological projects is that much of the early work is expected to be manually constructed knowledge bases.

Another area which, to a large extent, depends upon the manual construction of links and hypertext is the field of intelligent databases. As Parsaye, Chignell, Khoshafian, and Wong (1989) describe in their book, the defining attributes of intelligent databases are object-orientation, deduction and hypermedia (associative) technologies. They discuss machine learning in conjunction with intelligent databases, but it is machine learning as previously described. They do not describe how the construction of the associative hyperlinks can be automated.

Finally, this research uses what could be described as a somewhat connectionist architecture to represent its cases. There are precedents for this such as that described by Tirri, Kontkanen, and Myllymaki (1996) in their paper on instance-based learning. However, in the very first sentence of the abstract, the theme of using the acquired knowledge for prediction is mentioned. While there are similarities to this instance-based learning, their emphasis is on the ability to classify and predict using the joint probability distributions of the various attributes to define a class or concept. The knowledge base architecture which is automatically generated by our approach is presented in the following chapter.

CHAPTER 3

THE KNOWLEDGE BASE ARCHITECTURE

This work proposes a new approach to the acquisition of a declarative mental model or declarative associative memories. This section presents the architecture of the knowledge base which is the ultimate product of the knowledge acquisition. This architecture will serve as an introduction to the algorithm which creates the knowledge base from conventional databases and as an introduction to the associative retrieval algorithm. The design of all three components, the declarative memory structures, the learning algorithm, and the retrieval algorithm, was coordinated to ensure the proper functioning of the knowledge base as an associative memory/intelligent database. First, a conceptual view of the knowledge base is presented with an explanation of the motivation for each component and feature. Next, the details of the relational database implementation of the knowledge base is presented. The function of each table is described and related back to the conceptual architecture. As mentioned in the introduction, the ideas and motivation for this research came primarily from an interest in CBR and data mining for association rules. At the time that this research was conceived, other members of our research group were involved in the discovery of association rules in databases. The domain of that work was the detection of the development of antibiotic resistance in organisms which were of medical/public health significance (Brussette et al., in press). These association rules were associations of two or more values which were actually part of a larger incidence of resistance. Associations were considered to be interesting. For example, a certain

level of antibiotic resistance by a particular bacterium might be expected and known. This level was considered background noise rather than a signal indicating a problem. It is only when the change in the number of incidences of resistance over time (the slope) changes dramatically that epidemiologists would consider it to be a public health problem. The association rules which were discovered offered valuable indicators of developing antibiotic resistance but were not easily linked back to the underlying data.

Epidemiologists might want to view the underlying cases from which the association rules were derived. It was recognized that each record in the database from which the association rules were derived could be thought of as a case in the sense of CBR. One of the original ideas was to save these as summarized memories of past incidents of the development of antibiotic resistance or epidemics so that, when the kinetics of an individual association rule indicated a developing situation, it could be related back to a particular case (such as an epidemic) so that epidemiologists could realize what was happening before they had all the data. This would allow public health workers to take action in the early stages of the developing situation. We discovered, in fact, that a similar approach had been implemented (Bull, Kundt, & Gierl, 1997). Saving every association and relating each of these back to every record is computationally expensive and requires large amounts of secondary storage. Every association means every n -item association, where n is the number of items participating in the association. For an original record having five attributes, this might include all 2-item associations, all 3-item associations, all 4-item associations, and all 5-item associations. From these original ideas came the view that every attribute-value pair and record combination could be considered an association and that if the relative frequency of this association was included an association rule could be created. These association rules would capture the knowledge which

would, in human terms, reflect a person's most recent experience or their normal context. Consider the example of presenting a number of people with the word "screen" and asking for their most immediate associations. A computer user might respond with references to monitors or cathode ray tubes, and a chemist would most likely be reminded of the process of sorting particles by size ("screening"), whereas a carpenter might think of the metal mesh used to cover windows. In addition to discovering these associations, other objectives were to preserve the original data and to capture the semantic and class membership information which was implicit in the original data. In one sense this becomes an exercise in learning nouns by observing databases. The potential uses of the resulting knowledge base include as a persistent, cumulative memory for an intelligent system and as an intelligent, browsable database. As the knowledge acquisition algorithm was being developed, the interface as a memory for an intelligent system and the needs of an intelligent database were kept in mind and influenced the design of the learning algorithm. Actually, there are few differences in the use as a memory versus the use as an intelligent database. The intelligent database with its graphical user interface (GUI) provides a better demonstration, but the retrieval mechanisms would be the same. In the first instance items would be returned from memory and displayed by the GUI. One of the objectives of the intelligent database was to provide the ability to let the user browse the database with no prior knowledge of the structure (or even the content) of the database. In essence the user could approach the database/knowledge base with the question, "What do you know about S. aureus?" The database would then find those cases with which S. aureus was most frequently associated. These cases would be ranked and presented to the user. The user could then view other examples of bacterial infections, examine the attributes of bacterial infections or see other examples of cases

involving bacteria (not necessarily just infections). This interactive browsing can be restricted to the domain of bacterial infections or expanded to all domains. This example reflects reminding within a single domain, such as the current knowledge of a medical technologist. The relative frequency of associations of attribute =/value pairs such as the association of S. aureus with infections in certain demographic groups would be expected to change with time. When the medical technologist was first asked about infections involving a particular organism, the strongest reminding might be associated with children, whereas a year later the same organism might be more frequently associated with infections following surgery. The conceptual knowledge base design presented here can reflect these changes in the strength of associations. When used as a memory for an intelligent system, items returned from the knowledge base would be placed in the expected data structures rather than displayed by the GUI.

The Conceptual Knowledge Base

Conceptually, the model consists of nodes and associational and semantic links between these nodes. The strength of the links is dynamic. The calculation of these strengths is based on the relative frequency of a particular association compared to all other associations in which that item participates. The memory is persistent and cumulative. It will continue to learn with time. Multiple, diverse databases can be included so that the memory is knowledgeable about multiple domains. Table 1 presents example input data. It consists of nine records from a database of bacterial infections which might be encountered in a hospital laboratory.

Table 1

Example of Input Data

<u>Organism</u>	<u>Site of infection</u>	<u>Race</u>	<u>Gender</u>
<u>E. coli</u>	Liver	white	Male
<u>E. coli</u>	Liver	white	Male
Mycobacterium	Lung	white	Female
<u>S. aureus</u>	Ear	Asian	Male
<u>S. aureus</u>	Ear	Asian	Male
<u>S. aureus</u>	Bone	black	Male
<u>S. aureus</u>	Bone	black	Male
<u>S. aureus</u>	Bone	black	Male
<u>S. aureus</u>	Bone	black	Male

Figure 2 is a conceptual diagram of the associative memory created from Records 1 and 2 from the data in Table 1. The original records (duplicates combined) are represented as nodes. Attributes and values associated with each record are also represented as nodes. The arcs represent an association between each attribute/value and the individual cases. The numbers attached to each arc represent the relative strength of each association. The arcs linking the value nodes to the attribute nodes are designated as "IS_A" links.

Conceptually and physically, the knowledge base uses a distributed representation for cases. The item labeled "case 1" in Figure 2 holds only a case identifier. This node represents both Record 1 and Record 2 of the original input shown in Table 1. The real definition of the case is through the linking of this unique case identifier with the nodes representing the various attributes such as site of infection, gender, race, and organism and their corresponding values of liver, male, white, and E. coli.

The IS-A link between "case 1" and "infection" represents the set membership of that particular case in the set of infections. This set membership is derived from the name of the original input database table. This can be presented to the user simply as a

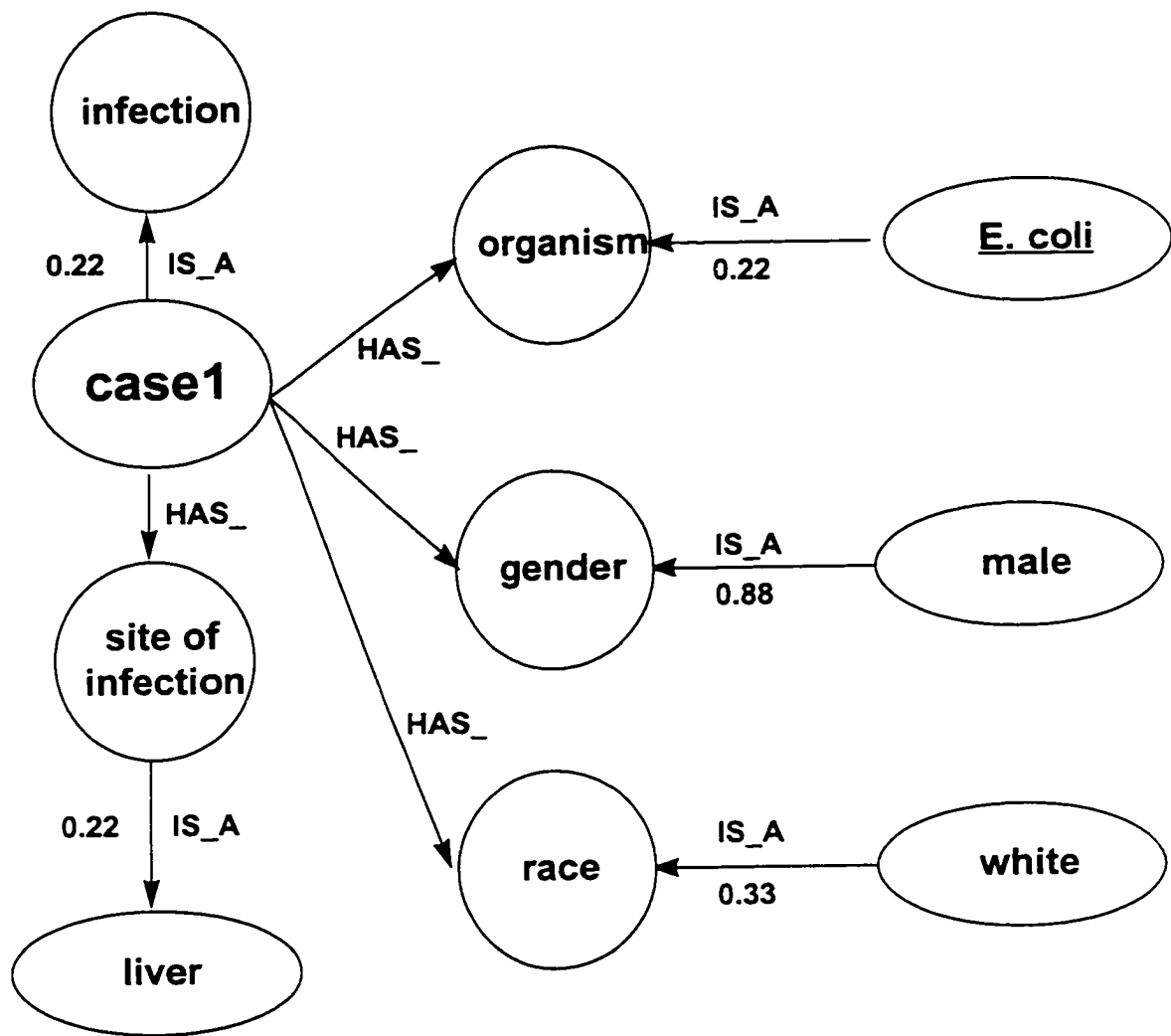


Figure 2. Conceptual view of the knowledge base.

fact or it can allow the user to request other examples from that particular set. The same is true for the other IS-A links shown in Figure 2. IS-A links serve to capture the implicit semantic information which was included in the original conventional database.

The links between the case identifier node and the values of liver, male, white, and E. coli can be interpreted as HAS-A links. For simplicity, the corresponding HAS-A links between the case identifier node and its attributes such as site of infection, gender, race and organism were not included in Figure 2. As discussed below in the section on the retrieval process, a case can be retrieved from an attribute, a value, or a combination of the two.

The strengths of the links are calculated by comparing the number of instances of a particular value to the total number of values. For example, there were nine records contained in the input table shown in Table 1. Of these nine records, two had E. coli as the value for the attribute organism. Therefore, $2/9$ yields a relative frequency of 0.22, which means that most of the time when organism is considered, E. coli would not come to mind. In contrast, S. aureus occurs as the value for organism. Its relative frequency is 0.77. As a final illustration, according to the input data, it is far more common for males to have bacterial infections than for females.

The Database Implementation of the Knowledge Base

The knowledge acquisition algorithm reads conventional relational databases as input and constructs the physical implementation of the conceptual design shown in Figure 2. This is done by populating tables in yet another relational database. Each of these tables implements a node and/or a link from the conceptual model. Tables 2-5 show the database tables resulting from the data mining process. The first of these is the CASE_

Table 2

The Case Base Table

Case ID	Attribute	Value	Value to Case Strength	Occurrences
1	ORGANISM	<u>E. coli</u>	1.0000	2
1	SITE OF IN-	liver	1.0000	2
1	RACE	white	0.6667	2
1	GENDER	male	0.2500	2
2	ORGANISM	Mycobacterium	1.0000	1
2	SITE OF IN-	lung	1.0000	1
2	RACE	white	0.3333	1
2	GENDER	female	1.0000	1
3	ORGANISM	<u>S. aureus</u>	0.3333	2
3	SITE OF IN-	ear	1.0000	2
3	RACE	Asian	1.0000	2
3	GENDER	male	0.2500	2
4	ORGANISM	<u>S. aureus</u>	0.6667	4
4	SITE OF IN-	bone	1.0000	4
4	RACE	black	1.0000	4
4	GENDER	male	0.5000	4

Table 3

IS_A_TABLE

Member	Strength	Set	Occurrences
<u>E. coli</u>	0.2222	ORGANISM	2
Liver	0.2222	SITE OF INFECTION	2
White	0.3333	RACE	3
Male	0.8889	GENDER	8
Mycobacterium	0.1111	ORGANISM	1
Lung	0.1111	SITE OF INFECTION	1
Female	0.1111	GENDER	1
<u>S. aureus</u>	0.6667	ORGANISM	6
Ear	0.2222	SITE OF INFECTION	2
Asian	0.2222	RACE	2
Bone	0.4444	SITE OF INFECTION	4
Black	0.4444	RACE	4

Table 4

CASE_IS_A_TABLE

Member	Strength	Set	Occurrences	Domain
1	0.2222222	INFECTION	2	BACTERIAL INFECTIONS
2	0.1111111	INFECTION	1	BACTERIAL INFECTIONS
3	0.2222222	INFECTION	2	BACTERIAL INFECTIONS
4	0.4444444	INFECTION	4	BACTERIAL INFECTIONS

Table 5

DB_ASSOCIATIONS_TABLE

DB	Value_To_Domain_Strength	Attribute	Value	Occurrences
BACTERIAL INFECTIONS	0.08	ORGANISM	<u>E. coli</u>	2
BACTERIAL INFECTIONS	0.08	SITE OF INFECTION	liver	2
BACTERIAL INFECTIONS	0.08	RACE	white	2
BACTERIAL INFECTIONS	0.08	GENDER	male	2
BACTERIAL INFECTIONS	0.04	ORGANISM	Mycobacterium	1
BACTERIAL INFECTIONS	0.04	SITE OF INFECTION	Lung	1
BACTERIAL INFECTIONS	0.04	GENDER	Female	1
BACTERIAL INFECTIONS	0.08	ORGANISM	<u>S. aureus</u>	2
BACTERIAL INFECTIONS	0.08	SITE OF INFECTION	ear	2
BACTERIAL INFECTIONS	0.08	RACE	Asian	2
BACTERIAL INFECTIONS	0.16	SITE OF INFECTION	bone	4
BACTERIAL INFECTIONS	0.16	RACE	Black	4

BASE_TABLE which contains a distributed representation of each case. Its structure and example data are presented in Table 2. The following is a detailed description of the purpose of each field of the CASE_BASE_TABLE. Each row (with duplicates incorporated) in the original database table becomes a case which is stored as a set of 5-tuples (one for each attribute-value pair) consisting of the following fields: (a) CASE ID--a unique identifier for each case, (b) ATTRIBUTE--the name of each attribute from the original database table, (c) VALUE--the corresponding value from the original database table, (d) OCCURRENCES--the number of times that this case appeared (duplicates) in the original database table, and (e) VALUE-TO-CASE-STRENGTH--the strength of association of this value to this case relative to the total number of associations with all other cases. This table implements the value-to-case associations. Thus, a case exists as a virtual entity consisting of multiple entries in the CASE_BASE_TABLE. Original records 1 and 2 were combined into CASE1. Following the conceptual example shown in Figure 1, CASE1 consists of four entries (one for each attribute-value pair), all of which have the common CASE_ID, "1". The VALUE_TO_CASE_STRENGTH shows the relative frequency of a particular value-to-case association compared to all other cases with which that value has an association. For example for CASE1, E. coli is the value for the attribute "ORGANISM." Because original records 1 and 2 were duplicates, E. coli occurred two times as the value for the attribute "ORGANISM". In the original database there were no other cases associated with the value E. coli. Thus, the probability of CASE1 being selected when E. coli is the input to the memory is 1.00.

The IS_A_TABLE defines the set membership of values to attributes. It allows a single value to potentially have membership in one or more sets (attributes). Table 3 shows the IS_A_TABLE resulting from the nine original records shown in Table 1.

The following is a detailed description of the purpose of each field of the IS_A_TABLE. Each attribute value pair from the original database is stored as a 4-tuple consisting of the following fields. This table captures the semantic information which is implicit in the original database schema. The IS_A_TABLE represents the set membership of the various values: (a) ATTRIBUTE--the name of each attribute from the original database table, (b) VALUE--the corresponding value from the original database table, (c) OCCURRENCES--the number of times that this attribute value pair appeared (duplicates) in the original database table, and (d) VALUE-TO-ATTRIBUTE-STRENGTH--the strength of association of this attribute to this value relative to the total number of associations with all other values. Consider the attribute, "ORGANISM". E. coli occurs as its value in two instances. There are seven other instances in which this attribute takes on other values. Thus, the probability that "ORGANISM" will have the value E. coli is $2/9$ or approximately 22%.

Table 4 is an example of the CASE_IS_A_TABLE. This table is similar to the IS_A_TABLE. It records the fact that each case is a member of a larger domain. The following is a detailed description of the purpose of each field of the CASE_IS_A_TABLE. Each case is stored (as a unique case identifier) to show its relationship to the original database table from which it was derived. Like the IS_A_TABLE, it captures the semantic information which is implicit in the database schema. This table stores the set membership information for each case. Entries in this table are 4-tuples consisting of the following fields: (a) Case ID--a unique identifier for each case, (b) Table Name--the name of the original database table, (c) occurrences--the number of times that this case appeared (duplicates) in the original database table, and (d) Case-to-Table-Strength--the strength of association of this case to this table relative to the total number of associations

with all other cases. In this example the domain of each case is bacterial infections.

Cases were placed in their own IS_A table to provide a cleaner design and to separate the calculation of case-to-domain strengths from that of the attribute/value strengths.

The final table is the DB_ASSOCIATIONS table shown in Table 5. This was created to store the relative frequencies of various values to the entire database or domain. For example if one mentioned the domain of bacterial infections, the associative memory should retrieve the most common values (in its experience) which are related to infections. In this example, if Table 5 represented the memory of a medical technologist, the strongest associations with bacterial infections are where the attribute "race" has the value "black" and where the attribute "site of infection" has the value "bone." The weakest associations are "Mycobacterium" as the value of "ORGANISM" and "female" as the value of "GENDER." As a further illustration of the intended use of this table consider the scenario where a colleague asked a medical technologist what type of infections were currently occurring in the hospital. The technologist would respond, "We are seeing quite a few S. aureus bone infections in black males."

The algorithm which builds this knowledge base is described in the following chapter.

CHAPTER 4

KNOWLEDGE ACQUISITION

A high level view of the algorithm is shown in Figure 3. The following pages describe this algorithm in increasing detail. Many details have been deferred.

Given the name and location of the input database, the first task of the algorithm is to read the database schema. From the schema a list of the database tables is created. Each table is processed until the list is exhausted. For each table, the schema is consulted to derive a list of attributes for that table. This list of attributes is stored in an `ATTRIBUTE_VALUE` array.

Next the total number of records in the current input table is counted. This count is retained to be used later in the calculation of the relative frequencies which are assigned as the strengths between nodes in the knowledge base.

For each attribute the corresponding value from the current record is stored in the `ATTRIBUTE_VALUE` array. The remaining preliminary step prior to actually making entries into the knowledge base is to count the number of input records which are duplicates of the current input record. This count is also retained to be used later in the calculation of the relative frequencies which are assigned as the strengths between nodes in the knowledge base.

The first step in creating a new case in the knowledge base is to make an entry for each attribute/value pair from the current input record in the `IS_A` table. As shown in Table 3, the `IS_A` table consists of four fields: the member, the set, the strength, and the number of occurrences. The set corresponds to the attributes of the current input record.

```

For each database
  Make a list of tables
  For each table
    Make a list of attributes
    Count No_of_Records
    For each Attribute-Value Pair
      Make an entry in the Attribute_Value_Array
    Next Attribute-Value Pair
    For each Record
      Make a list of values for this record
      Count the No_of_Duplicates
      For each Attribute_Value_Pair
        Add a record to the IS_A_TABLE
        Add a record to the DB_ASSOCIATIONS table
      Next Attribute
      Add a record to the CASE_BASE_TABLE
      Add a record to the CASE_IS_A_TABLE
      Delete all occurrences of this record from original table
    Next Record
  Next Table
  Recalculate IS_A_TABLE strengths
  Recalculate CASE_IS_TABLE strengths
  Recalculate DB_ASSOCIATIONS_TABLE strengths
Next Database

```

Figure 3. A high level view of the algorithm.

The member refers to the corresponding value for that attribute. The number of occurrences is the count of the number of duplicate records previously mentioned. The strength field represents the relative strength of the current value to its attribute compared to all other values for that attribute. Strength is calculated later using the number of occurrences. The purpose of making these entries in the IS_A table is to capture the semantic set-membership information which is implicit in the original database schema. This information is used later in the retrieval step to offer additional examples of members of a particular set.

The next step in creating the knowledge base is to make an entry in the DB_ASSOCIATIONS table as shown in Table 5. The purpose of this table is to capture the associations between each value and the domain (database name) from which it was acquired. As shown in Table 5, the DB_ASSOCIATIONS table consists of five fields: the database name, the attribute, the value, the value-to-domain-strength, and the number of occurrences. The number of occurrences is the count of the number of duplicate records previously mentioned. The strength field represents the relative strength of the current value to its domain compared to all other values for that attribute. Strength is calculated later using the number of occurrences.

At this point, if a case representing the current input record (duplicate records included) is not found in the knowledge base, the distributed representation is created in the CASE_BASE table. Each new case is assigned a unique case identification number which is calculated by incrementing the maximum current case number. In addition to the CASE_ID field, the CASE_BASE table has an attribute field, a value field, a value-to-case-strength field, and a number-of-occurrences field. To create the distributed representation for a case, a record is created for each attribute/value pair from the original

input record. Each of these entries in the CASE_BASE table is assigned the same unique case identification number. The strength field represents the relative strength of the current value to this particular case as compared to all other cases with which the value has an association. Strength is calculated later using the number of occurrences. The inclusion of the attributes in this table implements the HAS_A links which were described in the conceptual model. This association of values to cases and its relative strength are central to the retrieval process.

The final table involved in the population of the memory is the CASE_IS_A table. It serves the same function for cases that the IS_A table serves for individual values. It captures and makes available the semantic and set-membership information which is implicit in the original database schema concerning cases. The CASE_IS_A table consists of the fields member, set, strength, occurrences, and domain, where domain corresponds to the name of the table in the original input database. This information is used later in the retrieval step to offer additional examples of members of a particular set.

Following the entries into the various tables to create the knowledge base, the current input record (and any duplicates) is deleted from the input database. This is done to avoid erroneously entering that record again. Doing so would result in an inaccurate number of occurrences being recorded. This, in turn, would affect the calculation of the strength of the links between nodes in the memory. The process described above is repeated for each input record. When all input records have been processed, calls are made to three separate subroutines to recalculate the strengths in the IS_A table, the CASE_IS_A table, and the DB_ASSOCIATIONS table, respectively. These calculations were originally made each time a input record was processed; however, in the interest of computational efficiency, they were moved outside the record processing loop.

A More Detailed Description of the Algorithm

This section describes the details of some of the more important parts of the algorithm. Because this research was implemented using a relational database management system (MicroSoft's Access 2.0), all manipulations of both the input database and the output database (referred to as the case base, knowledge base or memory) were implemented as structured query language (SQL) queries.

Counting duplicate input records. After reading an input record, it must be determined if the record is unique among the input records or is one of several duplicate records. This is determined by the function COUNT_DUPLICATE_RECORDS, which accepts as parameters the name of the input database, the name of the table from which the current input record came, an array containing the attribute/value pairs for the current record, and a count of the number of attributes for the current input record. COUNT_DUPLICATE_RECORDS makes a call to CONSTRUCT_SIMILAR_RECORDS_QUERY, a subroutine which, using the parameters passed to it, constructs a query for duplicate records. COUNT_DUPLICATE_RECORDS makes an SQL call to execute this query and receives a count of duplicates of the current input record. This count is used by the overall algorithm as the previously mentioned "number of occurrences" to calculate the strengths of the various links as described in the conceptual knowledge base.

Determining if a case already exists. In the description of Schank's (1982) work, it was noted that only novel inputs caused the creation of new structures in memory. In this work it is also necessary to determine if a case already exists in the knowledge base.

If not, a new distributed representation is created for the new input record. Otherwise, the number of occurrences is simply updated. The function which determines if a case corresponding to the current input record already exists is called `IS_IT_ALREADY_IN_CASE_BASE`, which receives as parameters the database name, the number of attributes, and the attribute/value array for the input record.

The existence of a case which is identical to the input record is detected by determining if there is a case in the `CASE_BASE` table which has exactly the same attributes and values as the input record. Because cases have a distributed representation in the `CASE_BASE` table, this process requires that, for each attribute, a query must be constructed for that attribute and its corresponding value from the input record. `IS_IT_ALREADY_IN_CASE_BASE` is assisted in this process by the subroutine `CONSTRUCT_ONE_ATTRIBUTE_QUERY`, which constructs a query for each attribute/value pair from the current input record. These queries are executed, and for each record in the `CASE_BASE` table (where the record consists of a case id, an attribute, and its corresponding value) which matches the criteria, its score in the `CASE_SCORE` table is incremented. To ensure that only exact matches are returned, the total score is divided by the number of attributes. Only those cases with a score of 1.0 are considered exact matches.

Determining if an item already exists in the IS_A Table. It is also necessary to determine if an attribute/value pair already exists in the `IS_A` table. If the pair is not in the `IS_A` table, a new entry is made; otherwise, the number of occurrences for that pair is updated. The function `ALREADY_IN_IS_A_TABLE` determines which is required. This is accomplished by constructing and executing an SQL query which performs a

SELECT for all records which have the current attribute of the input record as the SET field in the IS_A table and its corresponding value in the MEMBER field. This is done for each attribute/value pair of the input record. The end result is that a particular attribute/value pair should appear only once in the IS_A table. All subsequent occurrences of that pair in input records will cause the number of occurrences field to be incremented, rather than creating a new entry.

Dynamic recalculation of the links. All input records are processed as described above. The number of occurrences of each item is noted in its respective table. However, the recalculation of the strengths of the various links is deferred until all input records have been processed by the main loop of the algorithm. This is a more efficient approach which eliminates the repeated recalculation of the same link strengths. These recalculations are done by four separate subroutines--one for each of four tables.

The first of these is RECALCULATE_VALUE_TO_DOMAIN_STRENGTHS. This subroutine first performs a query to make a list of all values in the DB_ASSOCIATIONS table. Then, for each of these values it determines the sum of the number of occurrences. For each individual record which contains the value, the number of occurrences (previously saved) is divided by the sum to derive the strength of that value in the input database relative to all other values found in the input database.

The strengths of the IS_A links, as in the example E. coli IS_A bacterium, is recalculated by the subroutine, RECALCULATE_IS_A_STRENGTHS. It does a similar calculation where the number of occurrences of a specific value for an attribute is divided by all occurrences of all values for that attribute. Updates are made to the IS_A table.

RECALCULATE_VALUE_TO_CASE_STRENGTHS calculates the relative strength of the links from individual values to cases. This strength is the relative strength of the link between this value and a particular case when compared to the links between that value and all other cases. This number indicates with which case a particular value is most strongly associated. This subroutine updates the strengths in the CASE_BASE table.

RECALCULATE_CASE_TO_DB_STRENGTHS performs an update of the strength of association of each case to the database table from which it was derived. To illustrate, within a database called "Bacterial Infections," there might be two tables, "Infections" and "Organisms." Given a number of records in the "Infections" table, this subroutine captures the implicit fact that each of these records is an example of an infection and the relative frequency of occurrence. If each record is unique, then all are equal examples of an infection. However, if some records are not unique and actually represent duplicates, then these would be more common examples of the set of infections. This relative frequency of occurrence is calculated by the subroutine and used later by the retrieval procedures. These updates are performed in the CASE_IS_A table.

An Example to Demonstrate the Knowledge Acquisition Algorithm

The following is a demonstration of the algorithm using specific input records as an example. Table 6 presents the example input records. This example assumes that the knowledge base is initially empty. First, it is determined that there are a total of three records in the input database. The construction of the knowledge base begins when the knowledge acquisition algorithm reads the first record. Next, the algorithm constructs and executes an SQL query to determine the number of records which are duplicates of the

Table 6

Input Records

Organism	Site_of_Infection	Race	Gender
<u>E. coli</u>	liver	white	female
<u>E. coli</u>	liver	white	female
<u>S. aureus</u>	bone	black	male

current record. In this example there is one duplicate where a white female has a liver infected by E. coli.

Table 7 shows the resulting CASE_BASE table. Records 1 and 2 are combined into Case 1. The “occurrences” field records the fact that there were two occurrences of this record. Case 2 represents the single remaining record. Notice that it has only one occurrence.

Table 7

The CASE_BASE Table

Case_ID	Attribute	Value	Value_to_Case_Strength	Occurrences
1	ORGANISM	<u>E. coli</u>	1.0000	2
1	SITE_OF_INFECTION	liver	1.0000	2
1	RACE	white	1.0000	2
1	GENDER	female	1.0000	2
2	ORGANISM	<u>S. aureus</u>	1.0000	1
2	SITE_OF_INFECTION	bone	1.0000	1
2	RACE	black	1.0000	1
2	GENDER	male	1.0000	1

In this example, each value has a value-to-case-strength of 1.0, which indicates that the particular value is associated exclusively with a single case. This strength is calculated by dividing the number of occurrences for a specific value/case by all occurrences for that value.

Table 8 presents the CASE_IS_A table, which was created from the three original records. This table records the fact that each of these cases belongs to the set of infections in the domain of bacterial infections. The domain of bacterial infections might include other sets such as “organism” and “antibiotics.” Case 1 represents two input records and therefore has an association to the set “INFECTION,” which is twice as strong as that for Case 2, which represents only one record. This strength is calculated by dividing the number of occurrences for a single case by the total number of cases in a given set.

Table 8.

The CASE_IS_A Table

Case_ID	Strength	Set	Occurrences	Domain
1	0.6666667	infection	2	BACTERIAL INFECTIONS
2	0.3333333	infection	1	BACTERIAL INFECTIONS

Table 9 presents the database association table. The function of this table is to record the relative number of occurrences of a single value to the total number of values found in the input database. This provides a means to preserve and retrieve the associations of values to a particular domain of expertise. In the current example, the values “female,” E. coli, “white,” and “liver” occur at twice the frequency as do the values “male,” S. aureus, “black,” and “bone.”

Table 9

DB_ASSOCIATION Table

Db	Attribute	Value	Value_to_Domain_Strength	Occurrences
BACTERIAL_INFECTIONS	GENDER	female	0.1667	2
BACTERIAL_INFECTIONS	GENDER	male	0.0833	1
BACTERIAL_INFECTIONS	ORGANISM	<u>E. coli</u>	0.1667	2
BACTERIAL_INFECTIONS	ORGANISM	<u>S. aureus</u>	0.0833	1
BACTERIAL_INFECTIONS	RACE	black	0.0833	1
BACTERIAL_INFECTIONS	RACE	white	0.1667	2
BACTERIAL_INFECTIONS	SITE_OF_INFECTION	bone	0.0833	1
BACTERIAL_INFECTIONS	SITE_OF_INFECTION	liver	0.1667	2

Table 10 presents a populated IS_A table, which implements a semantic hierarchy. Each value is used to populate the MEMBER field. Its membership in a particular set is noted by entries in the SET field. In this example, "bone" has been the site of infection 33% of the time as compared to "liver," which has been the site of infection 66% of the time. Similarly, "female," as the value for gender, is twice as common as "male."

All the strengths in these tables are dynamic. As new input records are converted to cases, the relative strengths may change to reflect the most recent observations of the knowledge base. Since the application of this algorithm to large databases is of practical interest, the issues of scalability of the algorithm are investigated in the next chapter.

Table 10

IS_A Table

Member	Set	Strength	Occurrences
black	RACE	0.3333	1
bone	SITE_OF_	0.3333	1
	INFECTION		
<u>E. coli</u>	ORGANISM	0.6667	2
female	GENDER	0.6667	2
liver	SITE_OF_	0.6667	2
	INFECTION		
male	GENDER	0.3333	1
<u>S. aureus</u>	ORGANISM	0.3333	1
white	RACE	0.6667	2

A Parallel Version of the Knowledge Acquisition Algorithm

One intent of this research was that the knowledge base was to be persistent and cumulative. That is, the system should continue to learn and should never forget any fact that it had previously learned. In addition, it was intended to learn from databases from various fields. If the process that has been described above can be considered to be the learning of nouns, then multiple databases would begin to address the brittleness problem which was previously mentioned. The system could simultaneously have knowledge of bacterial infections and soccer. The performance of the algorithm which has been described begins as a linear function and with time degrades to a low order polynomial ($O(n^x)$). Although this is quite tractable, for large databases time does become a factor. To compensate for this a parallel version of the algorithm was developed. Multiple copies of this algorithm can be run on separate processors. Each of these copies works from its own subset of the original input database. All copies write to a shared knowledge base. To prevent conflicts and the entry of duplicate items into the knowledge base, a form of database locking based on semaphores was implemented. This simple locking is

the only form of interprocess communication that is required. Figure 4 presents a high level view of the parallel version of the learning algorithm. The addition of the eight lines of code which are preceded by an asterisk is all that is required to convert the serial algorithm to a parallel algorithm. The case base (knowledge base) is implemented as a set of relational tables. This design decision allowed for a simple conversion to a parallel algorithm. The basic idea is that multiple processes could work simultaneously on the creation of a common case base because each could work on different tables at different times. The only requirement was a mechanism of locking and unlocking individual tables when a process was working on that particular table. As soon as a process was done with a table, the table was unlocked or released for use by another process. An additional table for system use was created to implement the locking mechanism. This LOCK table consists of only two fields. The first field contains the name of the table to be locked. The other field contains the name of the process which has a lock on the table. A unique database index was placed on the table field which prevents that table name from being entered again by causing an error condition. When this error condition occurs, the locking routine simply loops until it can successfully make the desired entry in the lock table. This looping allows one process to wait on another. In addition to the subroutine which locks certain tables by making entries into the LOCK table, there is also an UNLOCK subroutine. At various points in the algorithm, a process will finish with a database table and call the UNLOCK routine to release the table for use by other processes. The UNLOCK routine receives as parameters the name of the table to be released and the name of the calling process. By using the name of the calling process, errors in unlocking are avoided. A process can only unlock those tables which it has previously locked. A process cannot unlock another process's locks.

```

For each database
  Make a list of tables
  For each table
    Make a list of attributes
    Count No_of_Records
    For each Attribute-Value Pair
      Make an entry in the Attribute_Value_Array
    Next Attribute-Value Pair
    For each Record
      Make a list of values for this record
      Count the No_of_Duplicates
      * LOCK_A_TABLE("IS_A_TABLE", CASEBASE)
      * LOCK_A_TABLE("DB_ASSOCIATIONS", CASEBASE)
      For each Attribute_Value_Pair
        Add a record to the IS_A_TABLE
        Add a record to the DB_ASSOCIATIONS table
      Next Attribute
      *UNLOCK_A_TABLE("IS_A_TABLE", CASEBASE)

      *UNLOCK_A_TABLE("DB_ASSOCIATIONS",CASEBASE)
      *LOCK_A_TABLE("CASE_BASE_TABLE", CASEBASE)
      *LOCK_A_TABLE("CASE_SCORE_TABLE", CASEBASE)
      Add a record to the CASE_BASE_TABLE
      Add a record to the CASE_IS_A_TABLE
      * UNLOCK_A_TABLE("CASE_BASE_TABLE", CASEBASE)
      * UNLOCK_A_TABLE("CASE_SCORE_TABLE", CASEBASE)
    Delete all occurrences of this record from original table
  Next Record
  Next Table
  Recalculate IS_A_TABLE strengths
  Recalculate CASE_IS_TABLE strengths
  Recalculate DB_ASSOCIATIONS_TABLE strengths
Next Database

```

Figure 4. A high level view of the parallel algorithm.

CHAPTER 5

PERFORMANCE EXPERIMENTS

The algorithm was implemented in MicroSoft's Visual Basic 3.0 Professional Edition. All experiments were run on a personal computer utilizing two 200-Mhz Pentium processors. The operating system was MicroSoft's Windows NT 4.0. The parameters which were common to all experiments are listed below: (a) two 200-Mhz Intel Pentium processors; (b) 64 megabytes of random access memory; (c) Windows NT 4.0; (d) Windows NT multiprocessor kernel; (e) each process was run under its own copy of the NT DOS virtual machine running in a separate address space; (f) database management system: MicroSoft's Access 2.0.

The following experiments were run for both the serial and parallel algorithms: (a) five experiments, each of which adds an additional attribute/value the purpose being to characterize performance as a function of the number of attributes; and (b) an experiment in which 1,000 records are processed. Following this an additional 500 records were added to the input database, both the input and output databases were compacted, and processing resumed. The purpose of this experiment was to demonstrate the effect of database fragmentation on performance. Figure 5 presents the data in tabular form for both a single process and for two processes in which 1,000 records are processed. The results of these experiments are presented graphically in Figures 6-12. Each experiment processed 1,000 records. For those experiments with two processes, records were assigned as follows. Records 1-500 were assigned to the first process, while Records

Number of Attributes										
	1		2		3		4		5	
Records	1-P	2-P	1-P	2-P	1-P	2-P	1-P	2-P	1-P	2-P
100	34	22	39	26	39	26	48	27	49	31
200	73	46	84	53	84	54	100	56	104	61
300	118	73	134	83	134	84	157	87	164	97
400	168	103	190	116	190	116	219	120	228	130
500	224	137	252	149	252	151	286	156	298	166
600	285	171	319	185	319	188	359	198	373	205
700	354	210	397	223	397	230	443	238	456	245
800	439	253	491	264	491	272	546	282	552	289
900	549	297	607	309	607	318	668	330	674	335
1000	675	351	729	356	790	370	804	379	816	385

Figure 5. Comparison of one-process versus two-processes as a function of the number of attributes (time in seconds).

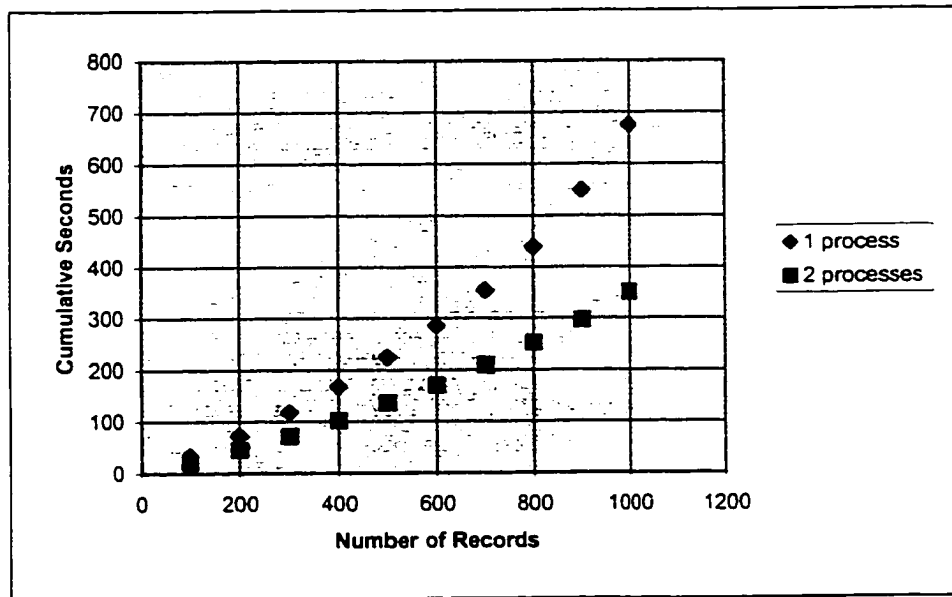


Figure 6. Comparison of serial and parallel versions for one attribute.

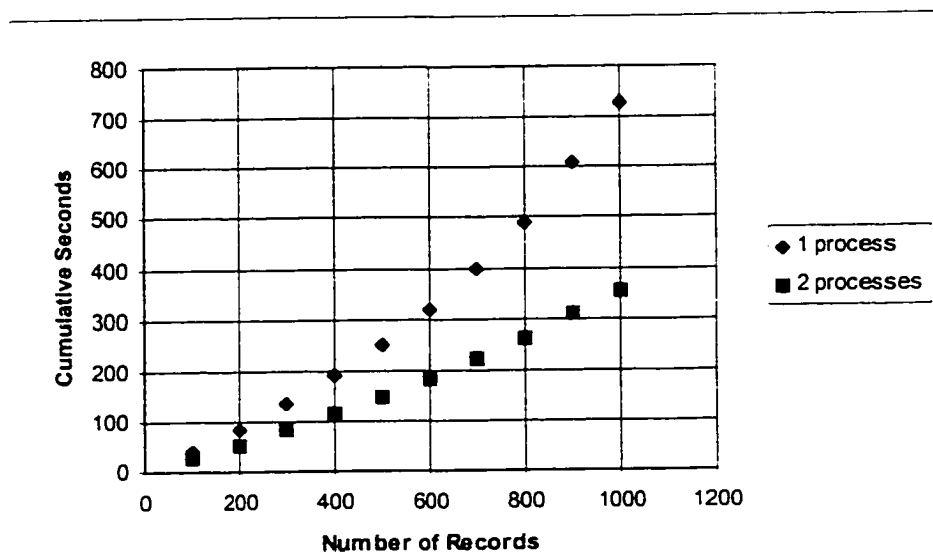


Figure 7. Comparison of serial and parallel versions for two attributes.

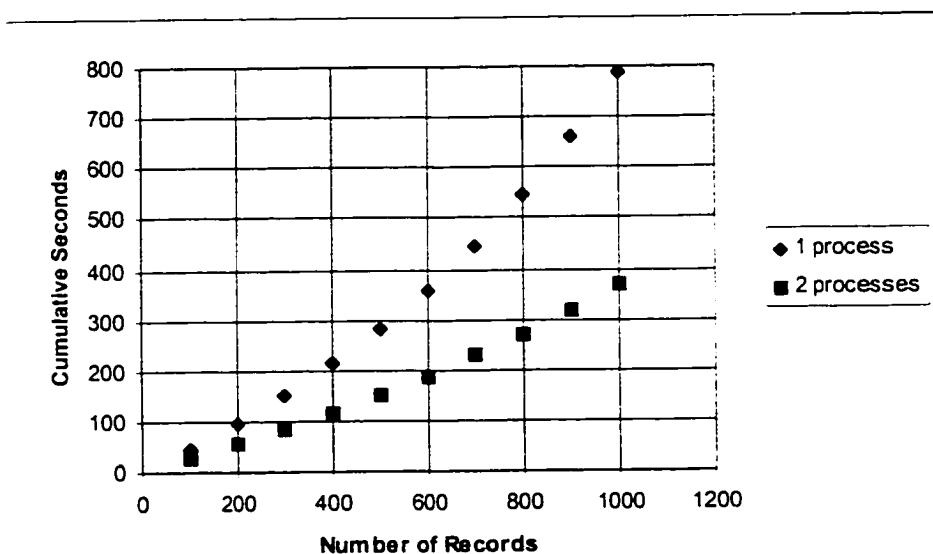


Figure 8. Comparison of serial and parallel versions for three attributes.

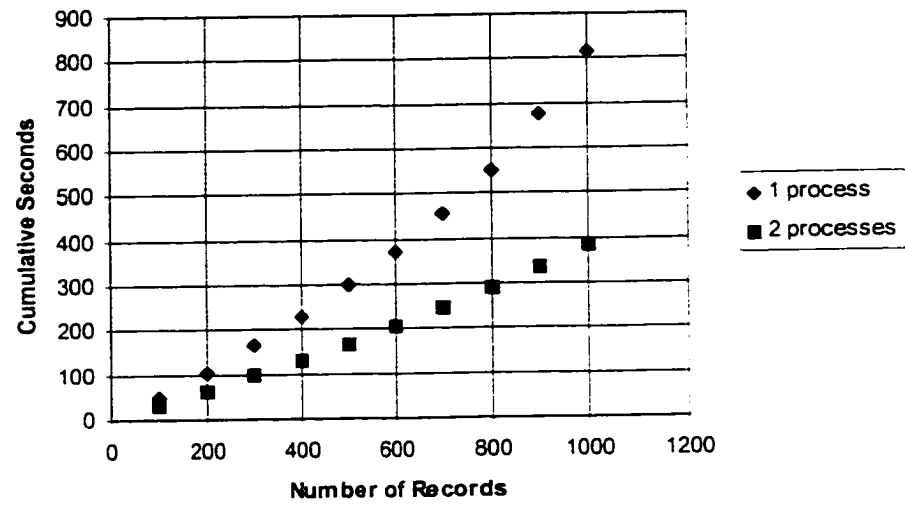


Figure 9. Comparison of serial and parallel versions for five attributes.

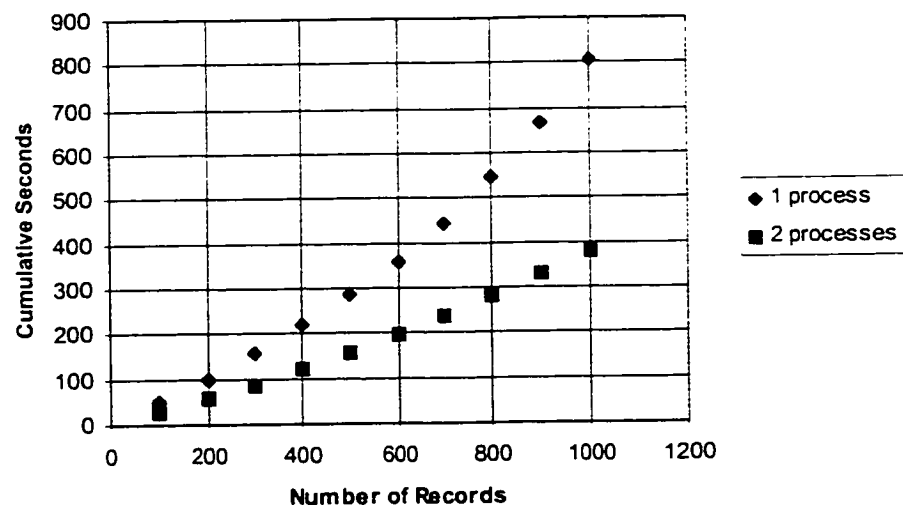


Figure 10. Comparison of serial and parallel versions for four attributes.

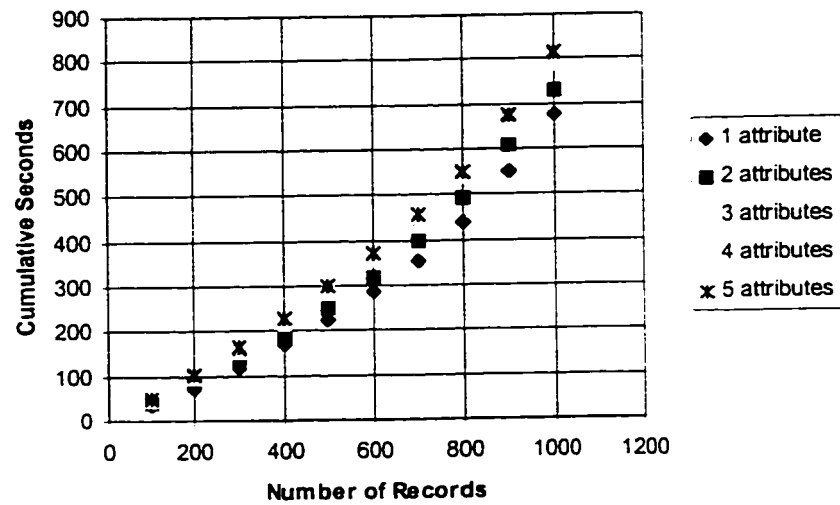


Figure 11. Performance of one process as a function of number of attributes.

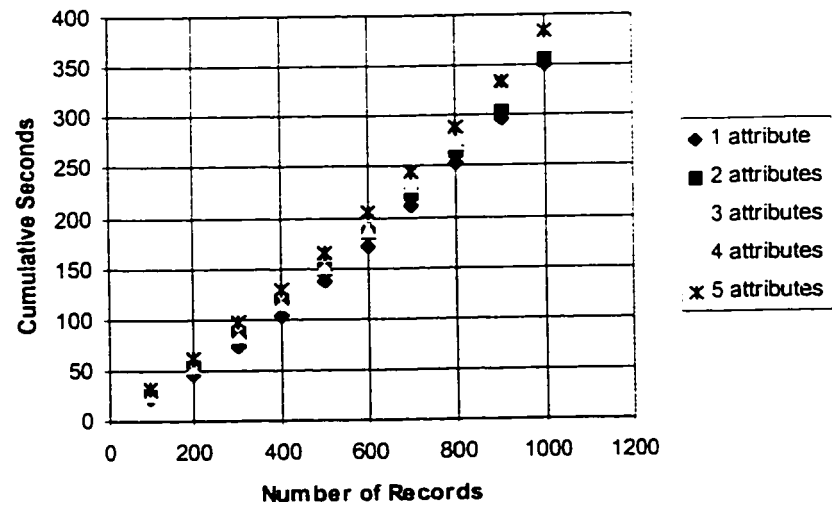


Figure 12. Performance of two processes as a function of number of attributes.

501-1,000 were assigned to the second process. Each record was unique. This series of 10 experiments show the effect of increasing the number of attributes from one to five. As shown by the table in Figure 6 and the graphs in Figures 7-12, the single process requires approximately twice the amount of time required by the two-process version. Both versions exhibit a basically linear response as a function of the number of input records. A linear regression of the data confirmed this. In all of the experiments the correlation coefficient was at least 0.98. The slope of the lines for the one-process version was approximately double that of the two-process version. In Figure 7, the slope of the one-process version was 0.68, while the slope of the two-process version was 0.36, giving a ratio of 1.88. As shown in Figure 10, where each input record had five attributes, the slope of the one-process version was 0.82, while the slope of the two-process version was 0.39, giving a ratio of 2.10. As the graphs in Figures 7-12 demonstrate, the one-process version shows a faster increase in curvature with an increasing number of input records.

As Figures 11 and 12 illustrate, the performance of both the one-process version and the two-process version is basically a linear function of the number of records. The number of attributes per record has very little effect on the time required to process an individual record. This is especially true of the two-process version.

The real source of performance degradation for this algorithm is the fragmentation of both the input and output databases. It is especially a problem for the input database. By using the debugging utility of Visual Basic™, it was determined that a major source of performance degradation was in counting the number of duplicates in the input database table for each input record and also in deleting these duplicates from the input database. Most other performance bottlenecks were corrected by the use of a number of indexes (in the database sense) on the various tables of the output database (the case

base) and by proper structuring of the algorithm. No attempt was made to index the input tables because it was assumed that their structure would not always be known to the user. One not-so-elegant solution to the problem of database fragmentation is to compact or reorganize the database periodically. Figure 13 presents the results of using MicroSoft's Access compact utility on both the input and output databases. This figure shows the time record to process 100 records as a function of cumulative total records processed. In this example, 1,000 unique records were processed with the resulting performance decrease shown in Figure 13. Following these 1,000 records, the process was stopped and both input and output databases were compacted. The process was then restarted to add an additional 500 unique records. This is shown as the abrupt drop in time required to process 100 records in Figure 13.

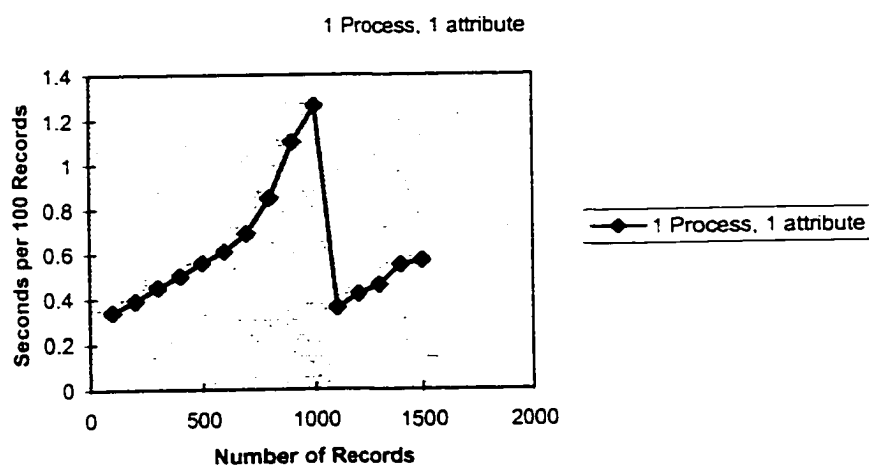


Figure 13. Effect of database reorganization on performance.

The experiments described thus far dealt only with the portion of the algorithm which makes the initial entries into the various case base tables. The four subroutines which then recalculate the strengths of the various links are run outside the main processing loop. The following experiment was run to determine the performance of these subroutines. Recalculations were run for 100-1,000 records in 100-record increments. Each record had one attribute. The results are shown in Figure 14, which makes the initial entries into the various case base tables.

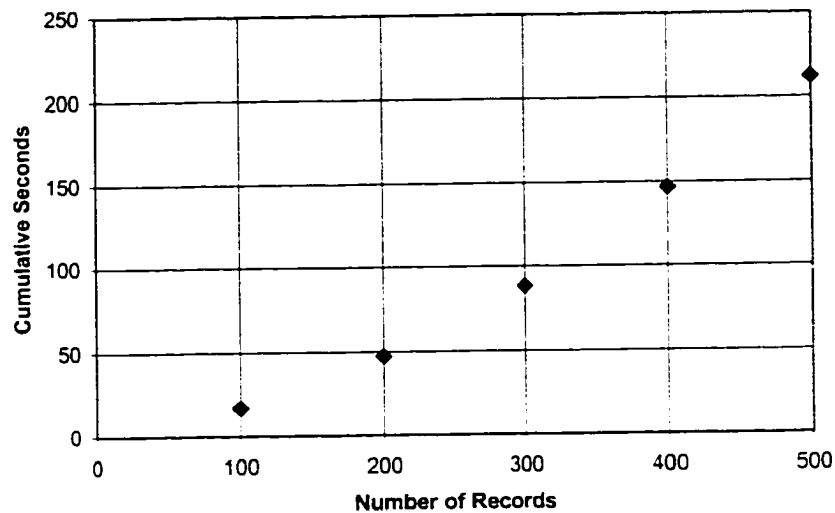


Figure 14. Performance of the strength calculation routines.

These experiments have demonstrated the scalability of the knowledge acquisition algorithm for generating the knowledge base. In the next chapter we describe the associative retrieval algorithm which provides user access to this knowledge.

CHAPTER 6

THE ASSOCIATIVE RETRIEVAL PROCESS

The ultimate purpose of all the work previously described was to construct and populate a memory from which items could be retrieved by both associative and semantic links. This section describes the procedure which was created to retrieve items from this memory. As stated earlier, this research can be considered from two viewpoints. The first of these is as the construction of a memory and a retrieval mechanism for an intelligent system with the intent that the intelligent system is learning about the world by observing the items in various databases. It is, in effect, learning nouns.

The alternate, although very closely related, view is as the conversion of conventional, passive databases into intelligent, active, interactive databases which take the initiative, return more than was asked, and have the ability to muse about their contents. This view as an intelligent, interactive database is used to describe the retrieval process.

The retrieval algorithm expects a memory which is structured as outlined in the description of the data mining algorithm. Interaction with the knowledge base begins with the user being presented with the screen shown in Figure 15. The user can enter words which describe the topic being searched for or can select items from the drop-down lists of DOMAINS, ATTRIBUTES, OR VALUES. The result is basically the same. The drop-down lists, in addition to providing a means of initiating a search, also are a form of metaknowledge which can be presented to the user. This metaknowledge is a list of the topics about which the system has knowledge. When input has been

PACK

INPUTS TO MEMORY

SEARCH CLEAR Text3 EXIT

TEXT

DOMAINS ATTRIBUTES VALUES

OUTPUTS FROM MEMORY

CASES	IS A	DOMAIN

USE CASE AS INPUT DISPLAY DETAILS RESTRICT DOMAIN

Figure 15. The initial screen.

specified and the SEARCH button is chosen, a call is made to the underlying subroutine ACTIVATE_ASSOCIATIONS. Conceptually, this routine activates the various items which are attached to cases as attribute/values. For example, referring back to Figure 3, E. coli is attached to Case 1 as a value of the attribute "organism." If the user enters E. coli, conceptually an activation flows from this value to Case 1. All other cases which have a link to E. coli are also activated. This activation of cases is implemented by incrementing that case's score in the CASE_SCORE table. For each attribute or value that matches exactly, a case's score is incremented by the strength of the link from that attribute/value to that particular case. The strength of the link from an attribute/value to a case was calculated by the knowledge acquisition algorithm as previously described. To provide a degree of fuzzy matching if no exact match is found, an attempt is then made to match on those attributes or values which have the input item as a prefix. If the input item is found as a prefix, then all cases which have a link with that attribute/value have their score incremented by 90% of the strength of the link from that attribute/value to that particular case. If both the exact match and the prefix match fail, then a substring

match is attempted. That is, a search is conducted for attributes/values which have the input item as an embedded substring. If a substring match is found, all cases which have a link with that attribute/value have their score incremented by 75% of the strength of the link from that attribute/value to that particular case. The exact match, prefix match, and substring match correspond to the $\text{sim}(f^I, f^R)$ component of the similarity metric of Figure 1. The values of 1.00, 0.90, and 0.75 correspond to the w^I or feature weighting factor in the similarity metric described in Figure 1. In this algorithm both the similarity function and the weighting factor are changed in an attempt to find partial matches. Incrementing case scores by 0.90 and 0.75 for prefix and substring matches, respectively, was an arbitrary decision. No attempt was made to devise more justifiable weights for partial matches. This process of matching against input terms and incrementing case scores is repeated for each input item. Queries are constructed for each of these searches by yet another subroutine(s). By restricting the CASE_BASE table to records containing only a single attribute/value combination searches, as described above can be conducted with the user (or calling program) having no knowledge of the structure or contents of the knowledge base or the original input databases.

A provision is also made to search for cases when the input item has been designated by the user as a domain name. If only the domain name is entered, then all cases which belong to that domain are retrieved from the CASE_IS_A table. Only exact matches of the domain name are attempted. Cases are returned in descending order of their "strength" field in the CASE_IS_A table. This strength was set by the data mining algorithm as previously described and represents the relative frequency of occurrence of this particular case relative to all other cases found in the same table of the input database.

If both a domain name and potential attribute/value terms are entered, the algorithm first matches on the attribute/value's and then on the domain name. Only those cases which match on the attribute/value's and are members of the specified domain will be returned. For matches of the attribute/value items, case scores are calculated using the attribute/value to case strength as described above. The domain matches do not use the case-to-domain strength. For each domain match each case score is incremented by a constant factor.

Cases are returned to the grid as shown in Figure 16. This grid consists of three fields--a CASE_ID field, an IS_A field, and a DOMAIN field. The CASE_ID field contains the unique identifier of the case retrieved from the CASE_BASE table. The contents of the IS_A field and the DOMAIN field are obtained by querying the CASE_IS_A table using the case identifier as a key. The IS_A field contains the name of the input table from which the case was constructed such as "Bacterial Infection." The DOMAIN field contains the name of the input database from which the case was originally acquired.

Having retrieved an initial set of cases, the user has several options. The first of these is to use one of the retrieved cases as input to find similar cases. This is similar to the concept of musing described by Fertig and Gerlenter (1988). Schank (1982) describes a similar idea in which two people are having a conversation. Each story or comment by one of the individuals reminds the other of yet another episode and so on, until they find that the topic of conversation had drifted far from the original. In the current work this musing or reminding is implemented by querying the CASE_BASE table for all attributes and values which are a part of the case selected by the user. This list of attributes and values is then used as input to construct additional queries for cases

PACK

INPUTS TO MEMORY

SEARCH **CLEAR** **EXIT**

TEXT

DOMAINS ATTRIBUTES VALUES

OUTPUTS FROM MEMORY

CASES	IS A	DOMAIN
CASE2	INFECTION	BACTERIAL INFECTIONS

USE CASE AS INPUT **DISPLAY DETAILS** **RESTRICT DOMAIN**

Figure 16. The initial search window.

which share these characteristics. The retrieved cases are not required to share all attributes/values with the original case. Any subset can lead to other cases.

A second option which is available to the user is to restrict the search to a particular domain. When a case in the grid has been highlighted and the “Restrict Domain” button has been toggled on, a variable, `GLOBAL_DOMAIN`, is assigned the value of the selected case’s domain. This is done because cases from various domains can have common attributes and values. When an initial search is done, the user may not be aware of this and may receive cases which are of no interest.

The third option available to the user when a set of cases has been retrieved is to select the “DISPLAY DETAILS” function. This function notes which of the columns in the grid have been selected. Based on this information it supplies the details of the case, the `IS_A` semantic links, or the domain. This process can be repeated as often as desired and allows yet another way to browse through the memory. Displaying the details of each of these items is described below. To illustrate these ideas the search begins with

the initial window shown in Figure 16. In this example, “Mycobacterium” has been entered as a search term. “CASE2” has been retrieved as the only case containing some form of the search term. The next two columns of the grid indicate that this case is an infection and that it came from the domain (input database) of bacterial infections. When the details of a case are requested, the case identifier is used to query the CASE_BASE table and retrieve the attributes and values of the case. The details of the case (including attributes and values) are displayed in a newly created window using Visual Basic’s Multiple Document Interface (MDI) capability, in which identical windows (each displaying different information) are created based on an original parent or prototype window. Continuing with the example from Figure 16, the details of CASE2 are displayed in Figure 17. Referring to Figure 17 (Case Details Window), the definition of the case is given by a listing of its attributes and their corresponding values. By selecting any of these attribute/value pairs and toggling the “More Values of This Attribute” button, the user can access the IS_A hierarchy which was created by the data mining algorithm. As an example for CASE2, the organism is Mycobacterium. If this attribute/value combination is selected, all examples of organism are displayed as shown in Figure 18. In this instance, the other examples of an organism are E. coli, Mycobacterium, and S. aureus. Once again, the traversal of memory can complete the cycle by requesting a search using any of these organisms as an input.

Referring again to Figure 17 (Case Details Window), the final path through memory from this window is by following the relationship links from case to case. These links were not constructed by the knowledge acquisition algorithm but were manually built into the memory. These links represent an area for future research. These

PACK

INPUTS TO MEMORY

SEARCH **CLEAR** **EXIT**

TEXT

DOMAINS ATTRIBUTES VALUES

OUTPUTS FROM MEMORY

CASES	IS A	DOMAIN
CASE2	INFECTION	BACTERIAL INFECTIONS

← →

USE CASE AS INPUT **DISPLAY DETAILS** **RESTRICT DOMAIN**

CASE2 - INFECTION

DOMAIN

Explore Domain

ATTRIBUTES/VALUES

ORGANISM - Mycobacterium

SITE OF INFECTION - lung

GENDER - male

More Values of This Attribute


RELATIONSHIPS

RESULTS IN → TUBERCLES(CASE5)




SIMULATE

Figure 17. The case details window.

INPUTS TO MEMORY

 **SEARCH** **CLEAR** **EXIT**

TEXT

DOMAINS  ATTRIBUTES  VALUES 

OUTPUTS FROM MEMORY

CASES	IS A	DOMAIN
CASE2	INFECTION	BACTERIAL INFECTIONS

◀ ▶

USE CASE AS INPUT **DISPLAY DETAILS** **RESTRICT DOMAIN**

CASE2 - INFECTION ▼ ▲

DOMAIN

Explore Domain

ATTRIBUTES/VALUES

ORGANISM - Mycobacterium ▲

SIT RELATIONSHIPS ▼ ▲

GE

Mo

ATTRIBUTE - ORGANISM ▼ ▲


<input type="text"/>	<input type="text" value="E. coli"/> <input type="text" value="Mycobacterium"/> <input type="text" value="S. aureus"/>	SEARCH
----------------------	--	---------------

Figure 18. Examples of IS_A_Organism.

relationship links are those other than the IS_A and HAS_A links which can be derived from the original data base schema. They are links such as “causes” or “results_in.” In the current example as shown in Figure 19 the first such link is a “Results_in” link from CASE2 (tuberculosis) to a case called CASE5 (Tubercles). That is, an infection by Mycobacterium can result in the formation of tubercles in the lungs. CASE5 (Tubercles) can in turn cause CASE6 (Pain). Each time the simulation option is chosen, a new MDI form displays the case which is the result of the relationship. In addition to displaying the details of cases, the retrieval process can directly display the details of the IS_A hierarchy and the domain. Either of these two items can be displayed by selecting the appropriate column from the grid and requesting a display of details. This allows the user to explore the IS_A hierarchy or the domain directly without having first to go through a specific case.

It was decided that any intelligent system/memory should possess a certain amount of short-term memory. That is, it should be able to remember its recent path through long term memory (the case base or knowledge base). To elaborate on this idea the previous discussion dealt with finding cases in long-term memory by searching for attributes/values or domains. Having found an interesting case, the system can then find similar cases and cases which are in the same domain or IS_A hierarchy. The path through memory consists of those cases which were interesting enough to open or display in detail. These cases are put into a data structure (array) which can be traversed in either direction. The items in this array consist of the following defined data type (Figure 20), where CASE_ID is used as a key into the CASE_BASE table for the reconstruction of the case from its distributed representation. The case is then displayed in an MDI window in the usual way. The integer items BACK_POINTER and FORWARD

INPUTS TO MEMORY



TEXT

DOMAINS

 ATTRIBUTES

 VALUES

OUTPUTS FROM MEMORY

CASES	IS A	DOMAIN
CASE2	INFECTION	BACTERIAL INFECTIONS

DOMAIN

ATTRIBUTES/VALUES

ORGANISM = Mycobacterium

SITE OF INFECTION = lung

GENDER = male

RELATIONSHIPS

RESULTS IN → TUBERCLES(CASE5)

DOMAIN

ATTRIBUTES/VALUES

SYMPTOM = TUBERCLE

RELATIONSHIPS

CAUSES → PAIN(CASE6)

Figure 19. Following relationship links.

```

Type STACK_ENTRY
  CASE_ID As Integer
  BACK_POINTER As Integer
  FORWARD_POINTER As Integer
End Type

```


Figure 20. CASE_STACK implementation.

_POINTER are indexes into the CASE_STACK array used by the system to remember its path through memory.

When a case is displayed in detail, an MDI window is opened. These windows can remain open, can be closed, or they can be minimized. This is at the discretion of the user. The ability to minimize case windows can be useful in saving interesting cases as the user browses through memory. The same ability applies to IS_A and domain detail display windows. Figure 21 shows an example of these minimized windows. Different (hopefully intuitive) icons were chosen for the different types of detail displays. A paper clip, a globe, and generic windows icons were chosen for cases, domains, and IS_A hierarchies, respectively. While not fundamental to the algorithms of this research, this feature hopefully will prove helpful to a user browsing the case base and will suggest the type of user interface for future work on intelligent databases.

PACK

INPUTS TO MEMORY



TEXT

DOMAINS

ATTRIBUTES


VALUES

OUTPUTS FROM MEMORY


CASES	IS A	DOMAIN
CASE1	INFECTION	BACTERIAL INFECTIONS

←
→


USE CASE AS INPUT
DISPLAY DETAILS
RESTRICT DOMAIN



CASE1 -
INFECTION



IS_A
INFECTION



DOMAIN -
BACTERIAL
INFECTIONS

Figure 21. Example of minimized detail display windows.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This research has presented an algorithm for the construction of a case base/knowledge base. The algorithm can be viewed in several ways. First, it can be thought of as supporting the data mining process. The result of data mining can be a set of association rules. However, in contrast to the conventional definition, we are not using association rules in the usual sense where two or more attribute/value combinations are found to be associated to some degree but with no contextual or explanatory information included. In the present work the association that is discovered and made available is the association between an attribute/value and the larger episode from which it was derived. These episodes are thought of as cases in the sense of CBR. The strengths of the associations are the relative frequency of the association between that attribute/value pair and any given case, relative to the strength of its association to all other cases. These strengths are intended to be dynamic so that as newer versions of the input database or databases from different domains are processed, the strengths of the links are recalculated to reflect the new inputs. This feature can be used to make the meaning of the association rules readily available and understandable.

We propose that the traditional data mining process described in chapter 2 be extended by the addition of a seventh step, described below.

1. Understand the problem.
2. Extract the data.

3. Clean/Engineer the data.
4. Engineer a data mining algorithm.
5. Search for potentially interesting patterns by running the data mining algorithm.
6. Evaluate the Patterns.
7. Retrospectively browse original data as episodal context for patterns of interest.

Addition of this Step 7 represents an enhancement of the data mining process and provides what Brachman and Anand (1996) have called “a human centered process” which includes a high level of human interaction. This view is captured by their definition of knowledge discovery: “Knowledge discovery is a knowledge-intensive task consisting of complex interactions, protracted over time, between a human and a (large) data base, possibly supported by a heterogeneous suite of tools.”

This research can also be viewed as the construction and population of a memory. It is a first step toward having computers learn about the world by the observation of databases. In a sense it can be viewed as the learning of nouns. The memory that is constructed is intentionally associative. As the algorithm was being developed, the ultimate purpose of associative retrieval was kept in mind and guided the design of the algorithm. In addition to capturing the associations which are explicit in the input databases, the algorithm also preserves and makes available the semantic information which is implicit in the input databases.

A view of this research which is closely related to the construction and population of an associative memory is the view that what the algorithm is actually doing is converting conventional databases into intelligent, interactive, proactive databases. In the

memory construction viewpoint, the intended user of the memory is an intelligent software system. In the intelligent database conversion viewpoint, the intended user is a human who queries and browses the database. With this in mind an associative retrieval mechanism with a GUI was implemented. This GUI demonstrates the capabilities of the system as an intelligent database. The retrieval mechanisms which are demonstrated would apply in exactly the same way if the memory is accessed by an intelligent software system.

This research suggests several directions for future work. Among these are the efforts to make the algorithm more efficient. As noted in the previous section, the main source of performance degradation is fragmentation of the input databases. This could be corrected to some extent by counting and eliminating duplicate records prior to processing the input. This might require the addition of a field for each record which could record the number of its duplicates. Another approach is the addition of indexes to each of the input fields so that direct access would replace database scans. There are also opportunities for further research in the use of parallelism. The experiments which were presented were conducted on a two-processor machine. An approximately twofold increase in efficiency was achieved by the use of two processors. Additional processors might result in greater efficiency. In addition to simply applying more processors to the entire algorithm, there are opportunities for dividing the algorithm into its functional parts and assigning each of these functions to a separate processor. Some preliminary work was done in which the CASE_BASE table and the CASE_IS_A table were populated separately from the IS_A table and the DB_ASSOCIATIONS table. Because there is no interference between these tables, there is no need for interprocess communication; therefore, multiple processors could prove to be extremely efficient. To accomplish this each

process is given its own duplicate copy of the input. This further eliminates any need for any interprocess communication.

The recalculation of the strengths of the various links is an obvious opportunity for pure parallelism. Each of the strengths is independent of all the others. Finally there is also an opportunity for research into the use of parallelism in the associative retrieval algorithm.

Concerning the construction of the memory an issue which could be addressed is the derivation of prototype cases rather than or in addition to the use of all individual instances. This is an area of CBR research which others are pursuing. Inferencing/simulation by the memory is a feature which would extend the original intent for this knowledge base. In its present form the memory is “reminded” of similar cases, other examples of the current item, and some relationships. It would be helpful if the memory could automatically perform all types of inferencing any time one of its items is activated. For example, in the introduction to this work it was noted that people instantly and without apparent effort know that things that come in contact with water become wet. This sort of background inferencing would make the system much more intelligent.

The system which has been described could very easily be adapted to document classification and retrieval. Having constructed a knowledge base from various domains, if a document abstract is used as input, a number of items in memory would be activated. When these items are returned as cases, their associated domains are also returned. The document could be classified into the most strongly activated domains. Having classified a number of documents, it could then recognize by the new input that, while a document search seems strongly associated with viruses, the user actually might be interested in

infections in general and not just viral infections. The system could then offer the user documents from the domain of infections.

If this research is viewed as the learning of nouns and concepts, then the most interesting direction for future research is machine learning by reading. This has been accomplished with quite good success by a number of researchers, but the limiting factor is the possession of enough common world knowledge to generate expectations which are then used to disambiguate word senses and to construct the internal representations of what has been read.

LIST OF REFERENCES

- Aamodt, A. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications 7(1), 39-59.
- Aha, D. W. (1991). Case-based learning algorithms. Proceedings of the Defense Advanced Research Projects Agency Case-Based Reasoning Workshop, pp.147-158.
- Bjork, E. L., and Bjork, R.A. (1996). Memory. New York & London: Academic Press.
- Brachman, R. J., & Anand, T. (1996). The process of knowledge discovery in databases. In U.M. Fayyad et al. (Eds.), Advances in knowledge discovery and data mining (pp. 37-57). Cambridge, MA: MIT Press.
- Brussette, S., Sprague, A. Hardin, J. M., Waiates, K. B., Jones, W. T., & Moser, S. A. (in press). Association rules and data mining in hospital infection control and public health surveillance. The American Medical Informatics Association.
- Bull, M., Kundt, G., & Gierl, L. (1997). An early warning system for detection and prediction of outbreaks of epidemics. Proceedings of GEOMED '97, 214-224.
- CBR Express for Windows User's Guide. (1990-1995). (p. 4). El Segundo, CA: Inference Corporation.
- Cohen, P., & Feigenbaum, E. (1981). The Handbook of Artificial Intelligence. Vol. 3. Los Altos, CA: William Kaufmann.
- Copeland, J. (1997). CYC: A case study in ontological engineering. Electronic Journal of Analytic Philosophy [On-Line]. Available: <http://www.phil.indiana.edu/ejap/copeland976.2.html>
- Epstein, H. (1997). Data mining: Exploiting the hidden trends in your data. DB2 Magazine, 2(1), 18-25.
- Eysenck, M.W., & Keane, M.T. (1995). Cognitive psychology: A student's handbook (3rd ed.). Hillsdale, NJ: Lawrence Erlbaum.
- Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. Communications of the Association for Computing Machinery 39(11) 27-34.

- Fertig, S., & Gerlinter, D. (1988). Musing in an expert database. Expert Database Systems Proceedings from the Second International Conference, 605 -620.
- Findler, N. V. (1979). A heuristic information retrieval system based on associative networks in associative networks. In N. V. Findler (Ed.), Representation and use of knowledge by computers (p. 307). New York: Academic Press.
- Ford, K. M., Bradshaw, J. M., Adams-Webber, J. R., & Agnew, N. M. (1993). Knowledge acquisition as a constructive modeling activity. In K. M. Ford & J. M. Bradshaw (Guest Eds.), [Special Issue] International Journal of Intelligent Systems Knowledge acquisition as modeling 8(1), Part I, 9-32.
- Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1991). Knowledge discovery in databases: An overview in knowledge discovery. In Databases (pp. 1-27). Menlo Park, CA: American Association for Artificial Intelligence Press.
- Goldstein, I. & Papert, S. (1977). Artificial intelligence, language, and the study of knowledge. Cognitive Science, 1(1), 84-123.
- Gomez, F., Hull, R., & Segami, C. (1994). Acquiring knowledge from encyclopedic texts. Institute for Electrical and Electronic Engineers International Journal of Artificial Intelligence Tools, 4(3), 349-367.
- Harris, Mary D. (1985). Representation of knowledge. In Introduction to natural language processing (p. 9). Reston, VA: Reston.
- Hill, A. E. (1987). An expert system for the diagnosis of malfunctions in a Kellogg ammonia plant. Unpublished master's thesis, Jackson State University, Jackson, MS.
- John, G. H. (1997). Enhancements to the data mining process. Unpublished doctoral dissertation, Stanford University, Palo Alto, CA.
- Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. Cognitive Science, 7(4), 243-280.
- Kolodner, J. L. (1993). What is case-based reasoning? In Case-Based reasoning (p. 4). San Mateo, CA: Morgan Kaufmann.
- Kolodner, J. L. (1996). Making the implicit explicit: Clarifying the principles of case-based reasoning. In D. B. Leake (Ed.), Cased-Based Reasoning, Experiences, Lessons, and Future Directions (pp. 349 – 370). Menlo Park, CA: AAAI Press.
- Kolodner, J. L., & Leake, D. B. (1996). A tutorial introduction to case-based reasoning. In D. B. Leake (Ed.), Cased-Based reasoning, experiences, lessons, and future directions (pp. 31–65). Menlo Park, CA: American Association for Artificial Intelligence Press.

- Kuipers, B. (1979). On representing commonsense knowledge. In N. V. Findler (Ed.), Associative networks, representation and use of knowledge by computers (p. 394). New York & London: Academic Press.
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. In P. S. Rosenbloom, J. E. Laird, & A. Newell (Eds.), The Soar Papers: Vol. 1. Cambridge, MA: MIT Press.
- Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. Communications of the Association for Computing Machinery, 38(11), 55-64.
- Leake, D. B. (1996). CBR in context: The present and future. In D. B. Leake (Ed.) Cased-Based reasoning, experiences, lessons, and future directions (pp. 3-30). Menlo Park, CA: American Association for Artificial Intelligence Press.
- Lenat, D. B., & Guha, R. V. (1990). Building large knowledge-based systems. Reading, MA: Addison-Wesley.
- Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. Communications of the Association for Computing Machinery, 19(3), 113-126.
- Parsaye, K., Chignell, M., Khoshafian, S., & Wong, H. (1989). Intelligent Databases. New York: John Wiley & Sons.
- Patil, R. S. (1981). Causal representation of patient illness for electrolyte and acid-base diagnosis. (Tech. Rep. No. 167) MIT Laboratory for Computer Science, Cambridge, MA.
- Porter, B., & Bareiss, R. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. Proceedings of the First International Meeting on Advances in Learning, Les Arcs, France, pp. 159-174.
- Quinlan, J. R. (1966). Semantic memory (Rep. No. AFCRL-66-189). Cambridge, MA: Bolt Beranek & Newman.
- Reisebeck, C. K. (1979). Representations to aid distributed understanding in a multiprogram system. In N. V. Findler, (Ed.), Associative networks, representation and use of knowledge by computers (p. 409). New York & London: Academic Press.
- Schank, R. C. (1973). Identification of the conceptualizations underlying natural language. In R. C. Schank & K. M. Colby (Eds.), Computer models of thought and language. San Francisco: Freeman.
- Schank, R. C., & Abelson, R. P. (1977). Scripts, plans, goals and understanding. Hillsdale, NJ: Lawrence Erlbaum.

- Schank, R. C. (1982). Dynamic memory. New York: Cambridge University Press.
- Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. Communications of the Association for Computing Machinery, 29(12), 1213-1228.
- Tirri, H., Kontkanen, P., & Myllymaki, P. (1996). Probabilistic instance-based learning. machine learning. In L. Saitta (Ed.), Proceedings of the Thirteenth International Conference. San Francisco: Morgan Kaufmann.
- Wettschereck, D., Aha, D. W., & Mohri, T. (1997). A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms in artificial intelligence review. Artificial Intelligence Review, 11, 273-314.
- Wu, X. (1995). Knowledge acquisition from databases. Norwood, NJ: Ablex.

**GRADUATE SCHOOL
UNIVERSITY OF ALABAMA AT BIRMINGHAM
DISSERTATION APPROVAL FORM
DOCTOR OF PHILOSOPHY**

Name of Candidate Aubrey Hill

Major Subject Computer and Information Sciences

Title of Dissertation Automated Knowledge Acquisition of Case-Based Semantic

Networks for Interactive Enhancement of the Data Mining Process

I certify that I have read this document and examined the student regarding its content. In my opinion, this dissertation conforms to acceptable standards of scholarly presentation and is adequate in scope and quality, and the attainments of this student are such that he may be recommended for the degree of Doctor of Philosophy.

Dissertation Committee:

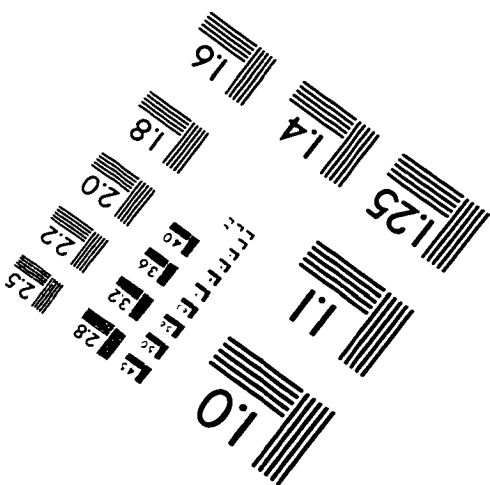
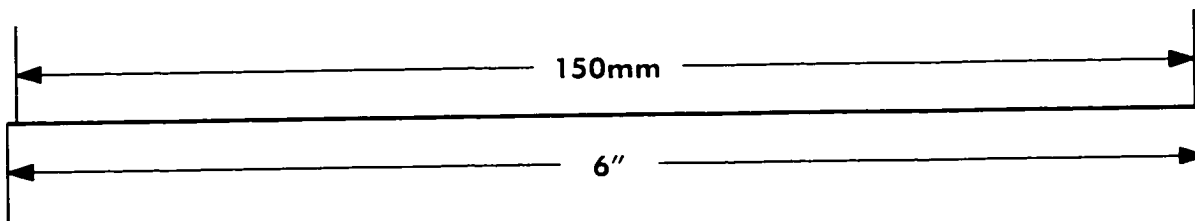
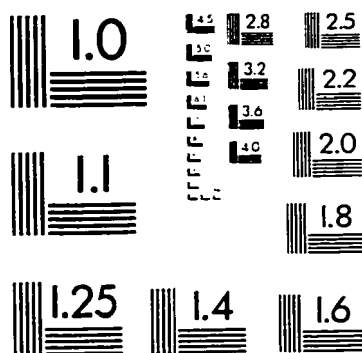
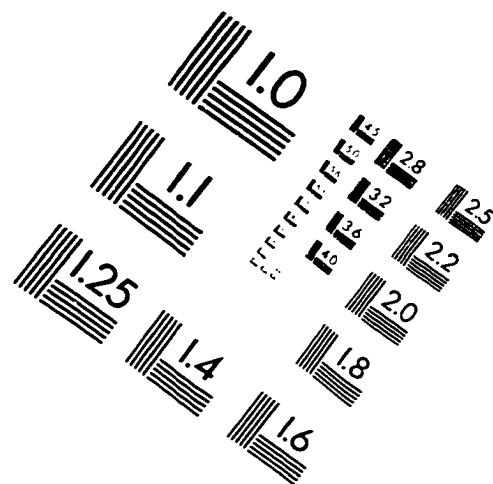
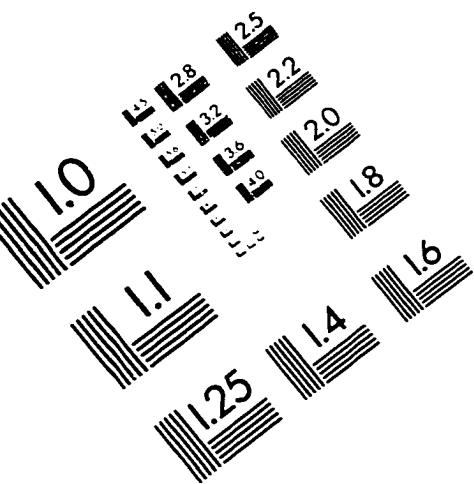
Name	Signature
<u>Dr. Warren Jones</u> , Chair	<u>Warren J. Jones</u>
<u>Dr. Robert Hyatt</u>	<u>Robert M. Hyatt</u>
<u>Dr. Barrett Bryant</u>	<u>Barrett Bryant</u>
<u>Dr. J. Michael Hardin</u>	<u>J. Michael Hardin</u>
<u>Dr. Sanjay Singh</u>	<u>Sanjay M. Singh</u>
<u> </u>	<u> </u>

Director of Graduate Program Warren J. Jones

Dean, UAB Graduate School Joan L. Linder

Date 7/6/98

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved

