12-1996

Utilizing Data and Knowledge Mining for Probabilistic Knowledge Bases

Daniel J. Stein III

UTILIZING DATA AND KNOWLEDGE MINING FOR PROBABILISTIC KNOWLEDGE BASES

THESIS

Daniel Joseph Stein III
Captain

AFIT/GCS/ENG/96D-25

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/96D-25

UTILIZING DATA AND KNOWLEDGE MINING FOR PROBABILISTIC KNOWLEDGE BASES

THESIS

Daniel Joseph Stein III
Captain

AFIT/GCS/ENG/96D-25

AFIT/GCS/ENG/96D-25

UTILIZING DATA AND KNOWLEDGE MINING FOR PROBABILISTIC KNOWLEDGE BASES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Science

Daniel Joseph Stein III, B.S.

Captain

December, 1996

Acknowledgments

There are many individuals who made this experience a little more bearable for me, without whom this would have truly been an ordeal. Unfortunately, one page is not enough to properly thank each of them.

First, I'd like to thank God for bringing me here and for helping me through the journey. His gift of a beautiful baby girl, Brittany, was just what I needed to keep the stress level down and to maintain my sanity.

Thanks to my advisor, Maj Sheila Banks, for her insightful comments and helpful suggestions. Thanks to Dr. Eugene Santos for keeping me focused throughout the process. Thanks to Dr. Henry Potoczny for his quiet tolerance and for generously giving his time and advice. And last, but not least, thanks to Capt Michael Talbert for continuously encouraging and sustaining me. His insights and ideas were invaluable. I'll miss the "30-second" discussions tremendously.

The AFIT library staff is positively "first rate." They are an exceptional group of individuals who truly go "above and beyond the call."

Once in a great while, you encounter an individual who gives more than the job description requires. Pat White is one of those people. She was there lending me a hand throughout the research process. Having her assistance with my research was like having a silent partner. On more than one occasion, she located a valuable resource for me long after I'd left the library. Thank you *very* much, Pat, for everything.

My dear wife Tamara deserves an award simply for tolerating me these last 18 months. This is the second time she has been with me during a degree program. It wasn't easy the first time, either. She said, "For better or worse," and she's proven that she meant it. I love you, darling.

Trust in the Lord with all your heart, and do not lean on your own understanding. In all your ways acknowledge Him, and He will make your paths straight. Proverbs 3:5-6

Table of Contents

Page

List of Figures

Figure Page

List of Tables

AFIT/GCS/ENG/96D-25

Abstract

Problems can arise whenever inferencing is attempted on a knowledge base that is incomplete. Our work shows that data mining techniques can be applied to fill in incomplete areas in Bayesian Knowledge Bases (BKBs), as well as in other knowledge-based systems utilizing probabilistic representations. The problem of inconsistency in BKBs has been addressed in previous work, where reinforcement learning techniques from neural networks were applied. However, the issue of automatically solving incompleteness in BKBs has yet to be addressed. Presently, incompleteness in BKBs is repaired through the application of traditional knowledge acquisition techniques. We show how association rules can be extracted from databases in order to replace excluded information and express missing relationships. A methodology for incorporating those results while maintaining a consistent knowledge base is also included.

UTILIZING DATA AND KNOWLEDGE MINING FOR PROBABILISTIC KNOWLEDGE BASES

I. Introduction

In this research we present a methodology and a tool for mining association rules and incorporating those rules into a knowledge base. The purpose of this is to repair knowledge base incompleteness uncovered during validation. We will discuss how this incompleteness is uncovered and show the three fundamental forms this incompleteness can take. We then describe how the mining of association rules are used to solve the incompleteness. We also show how we focus traditional techniques in order to make data mining operations more tractable.

1.1 Problem Statement

The relationship between the efficiency of a reasoning algorithm and the flexibility of its knowledge representation scheme is an inverse one. In order to implement a realistic, real-world application, both of these properties must not only be balanced, but maximized as well. Bayesian Knowledge Bases (BKBs) provide the needed blend of efficiency and flexibility, while also providing an ease of understanding lacking in many representation schemes [19]. For this reason, we have chosen the BKB representation for our *Probabilities, Expert System, Knowledge, and Inference* (PESKI), an integrated framework for expert system development [19].

Validation of a BKB is performed by submitting test cases and comparing the expected solutions with the actual ones. Incompleteness in a BKB is encountered during testing whenever the inference engine cannot reach one or more elements in the expected solution. This normally happens because one or more relationships are missing from the BKB. This work addresses a method for automatically extracting the necessary relationships uncovered during testing and incorporating those relationships back into the BKB.

1.2 Goals of This Research

The problem of inconsistency in BKBs was addressed by Gleason [6], where reinforcement learning techniques from neural networks were applied. However, the issue of automatically solving incompleteness in BKBs has yet to be addressed. Presently, incompleteness in BKBs is repaired by applying the same knowledge acquisition techniques that created the BKB. Whenever incompleteness is encountered, the PESKI user must manually augment the BKB to fill in the missing areas. In this work, we show that it is possible to repair each of the primary forms of incompleteness[1] using data mining techniques. Typical data mining approaches can become bogged down by an overabundance of patterns. In this research, we show how our approach, *goal-directed* data mining, can help bound the scope of data mining operations and make those operations more feasible.

[1] See Chapter 2 for a more thorough discussion of the forms of incompleteness.

1.3 Thesis Overview

In Chapter 2, we provide a thorough discussion of the problem background and describe some of the existing data mining techniques that were considered for the issue at hand. Chapter 3 covers the methods we considered and discarded, and then describes how we utilized data mining to extract probabilistic information from existing databases in a form suitable for incorporation into a BKB (and also into any other probabilistic knowledge base). A methodology for incorporating the data mining results is contained in Chapter 4. Chapter 5 discusses the actual implementation of the PESKI data mining tool and its integration into the PESKI framework. Conclusions and recommendations for further research are included in Chapter 6.

II. Background

This research sprang from a need to address incompleteness discovered during knowledge base validation. Here we present background material pertinent to our effort. We briefly discuss knowledge acquisition as a whole, and the knowledge representation scheme for PESKI, the Bayesian Knowledge Base (BKB). We then discuss verification and validation in PESKI, how validation can uncover incompleteness, and the fundamental ways incompleteness presents itself in a BKB. We then present a background discussion in data mining and a number of methods we considered for repairing incompleteness.

2.1 The PESKI System

PESKI is the physical realization of an integrated knowledge-based system framework that combines the functions of natural language interface, inferencing, explanation and interpretation, and knowledge acquisition and maintenance into a single, consolidated application [6], [20]. PESKI is the combination of the following closely interrelated, yet specialized tools:

1. Intelligent Graphical User Interface
2. Inference Engine
3. Knowledge Acquisition
4. Verification and Validation
5. Data Mining

PESKI is presently in the late prototype stage, where each of the fully functioning components are being brought together into a single cohesive whole. The interoperability of the separate components within PESKI, and how the data mining tool contributes to the overall project, will be addressed in detail in Chapter 5.

2.2 Knowledge Acquisition and Representation

Knowledge acquisition is defined as the transfer and transformation of problem-solving expertise from a knowledge source (usually a human expert) to an automated system. The knowledge acquisition phase represents a significant portion of the development effort for a knowledge-based system, and normally involves face-to-face interviews between one or more domain experts and one or more knowledge engineers. It has been called the bottleneck of artificial intelligence and is arguably the most difficult phase in the development of knowledge-based systems [9], [7].

The process of knowledge acquisition is very tightly coupled with that of knowledge representation. A knowledge representation scheme suitable for the problem domain is crucial to the successful completion of the overall system. In general, a successful acquisition strategy can be negated by poor or inadequate representation and vice versa. Unfortunately, a poor choice for a representation scheme is not always apparent. The knowledge engineer may progress all the way through system development before realizing the inferiority of the representation scheme, thus having to re-implement the system [7].

2.3 The Bayesian Knowledge Base Representation

BKBs depend on Bayesian probabilities to represent uncertain information in a knowledge base. This probabilistic aspect of BKBs make them almost ideal for operating in an uncertain environment [19]. Figure 2-1, which represents a single piece of information in BKB format, illustrates the structure of a BKB. Read in English, it would say, "Given that it is raining, it is almost certain that the sidewalk is wet." By BKB convention, the term "almost certain" would represent the probabilistic range[2] of 0.90 - 1.00. The large ovals are called *instantiation nodes*, or I-nodes for short, and represent a single state of a component. In mathematical terms, a component would represent a random variable and an I-node would represent a single instance of a random variable. The solid black circles are *support nodes*, or S-nodes, and represent the probability associated with one or more I-nodes. The support conditions for a given I-node consists of all the other instantiation nodes belonging to that I-node's direct ancestry. In Figure 2-1, the support for I-node Sidewalk = Wet consists of the single I-node Weather = Raining.

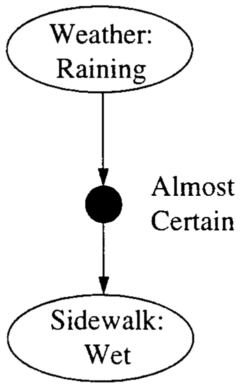


Figure 2-1

[2] See Santos [19] for each of the probabilistic ranges used in BKBs.

The theoretical foundation of BKBs is strongly grounded in probability theory. This probabilistic framework also enables BKB systems to make inferences using incomplete knowledge. Incompleteness is allowed in a BKB, but only as long as the requirements (i.e., the conclusions drawn based on given evidence) of the BKB are kept consistent. Incompleteness in BKBs occurs whenever essential connections are missing between I-nodes or when I-nodes lack certain necessary states[3]. Problems can arise whenever inferencing is attempted on a knowledge base which is incomplete. Our work shows that data mining techniques can be applied to fill in incomplete areas in BKBs, as well as in other knowledge-based systems utilizing probabilistic representations.

2.4 Knowledge Base Verification and Validation

Once the knowledge acquisition process is assumed to be complete, the knowledge base must be validated and verified by the knowledge engineer and/or the problem domain expert. There are a number of different types of errors that could be present in a knowledge base, but the focus of this research is on solving incompleteness that was discovered during verification and validation by applying data mining techniques.

The ultimate purpose of verification and validation is to guarantee that a knowledge-based system works properly and produces the correct answer in the correct form. Verification is "building the system right," a demonstration of the consistency and completeness of a system with respect to its specifications. In short, it confirms that the system actually does what the

[3] Incompleteness is evident when some answer in a test case cannot be concluded based on the evidence in that test case. See Section 2.5 for more details on test cases. See Gleason [4] and Lyle [9] for a more thorough discussion of incompleteness in BKBs.

specifications say it will do. It also ensures that the system is free from semantic and syntactic errors. Validation refers to the process of "building the right system." It involves insuring the system is correct with respect to user needs and system requirements. Validation guarantees the system produces the correct output and that it does what the users actually want it to do [7], [14].

2.5 Incompleteness Uncovered During Verification and Validation

Validation is performed in PESKI by submitting a series of test cases and comparing the resulting solution to the one that was expected. Each test case consists of a number of pieces of evidence (known events) and a set of expected answers (anticipated events). A test case is said to be valid if the expected answers are part of the overall solution set obtained after inferencing over the BKB based on the evidence. Normally, if the submitted answers aren't present, the weights inside our network are adjusted by applying reinforcement learning, to force the answers into our solution set [6].

Incompleteness in a BKB cannot be determined simply by inspection. Contrarily, it must be determined based on test cases and the semantics of the knowledge base. In PESKI, incompleteness is evident whenever it is impossible to conclude one or more elements of the answer set given the evidence. Incompleteness uncovered during validation can occur in one of three fundamental ways. It occurs when

1. A relationship between two different states in the BKB is missing. In other words, there may not be a probabilistic link, either direct or indirect, between some state in the evidence set and some other state in the answer set.
2. A given state may have insufficient support conditions (i.e. more evidence is needed to indicate the instantiation of a given state).

3. A component has one or more missing or unspecified states. For example, the component Weather might have states Cloudy, Foggy, and Rainy present, but the state Snowing might be missing.

Figure 2-2 and Figure 2-3 illustrate one example of incompleteness in a BKB[4]. The knowledge base shown in Figure 2-2 depicts the state of a Sidewalk based on Sprinkler and Weather activity. On the surface, the BKB appears to be complete, since the obvious relationships are captured.

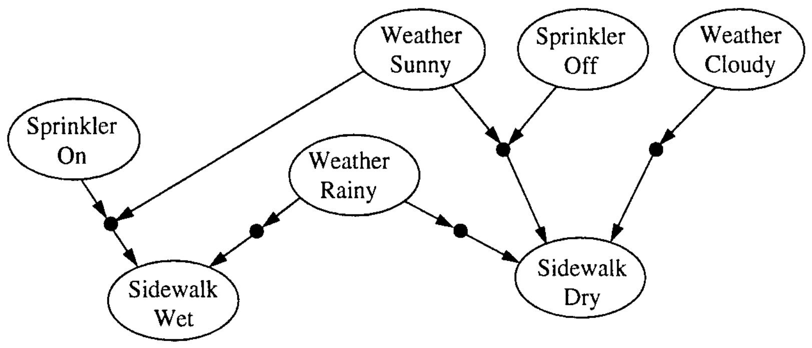


Figure 2-2

Now observe the answer and answer nodes as shown in Figure 2-3. Assume for a moment that the sprinkler system is state-of-the-art and that it is designed to shut itself off whenever it is raining. If this were true, then we would want to know the probabilistic relationship between Weather = Rainy and Sprinkler = Off and the test case depicted in Figure 2-3 would be meaningful to the user. However, since the answer doesn't directly depend[5] on the evidence, the knowledge base is said to be incomplete.

[4] The probabilistic values have been intentionally left out to keep the figures uncluttered.
[5] See Lyle [11] for a more thorough discussion of test cases and dependency regions.

We consider data mining to be a promising method for repairing incompleteness in BKBs. In Section 2.6, we discuss some of the most promising data mining techniques and show how each may apply to repairing incompleteness.

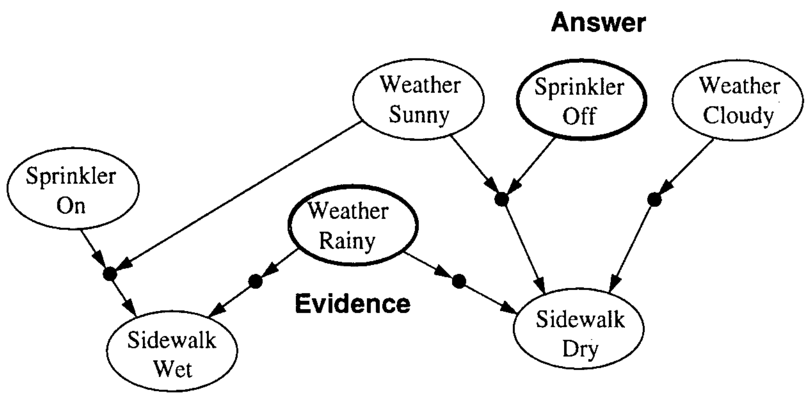


Figure 2-3

2.6 Data Mining

Even though machine learning, a close relative of data mining, has been researched for a number of decades, the field of data mining is still relatively immature. Significant work in the field of data mining has been accomplished only in the last few years. It has only been in the last eight or nine years that data mining has been distinguishable from other knowledge gathering activities [5] , [13].

Attempts at trying to define data mining usually result in circular arguments about the meaning of *data*, *knowledge*, and *information* and what it means to learn. A good working definition for data mining is *the automatic extraction of useful information from raw data* [22].

Data mining usually refers to tools and methods used to extract meaningful information from data that is unformatted and either unstructured or partially structured [10].

There are a number of different synonyms for data mining, including knowledge extraction, database exploration, information harvesting, and knowledge discovery in databases (KDD). In each case, the purpose of the activity is the nontrivial extraction of implicit, previously unknown, and potentially useful information. Some of the different data mining activities include information extraction, clustering, data summarization, extracting association rules, learning classification rules, and finding dependency networks [16]. In this section we present a brief overview of the most relevant data mining techniques considered for this work and some of the key features of each technique. In Sections 2.6.4 through 2.6.7, we also discuss some general data mining issues confronting the developer of a data mining tool.

2.6.1 Information Extraction

Information extraction systems attempt to obtain meaningful information from some text-based source, such as an on-line encyclopedia or an Internet web page. Most often, these systems are highly focused and specialize in a single application or domain area. They usually deal only with specific types of text and are only partially accurate. In some applications of information extraction, a high-precision text classifier is employed to determine the relevancy of a given text prior to actual mining efforts begin [18]. According to Cowie and Lehnert [4], a slightly different type of method for acquiring information from text, *knowledge* extraction, is being performed at some research sites. The goal of these systems is to deduce a rule base or domain model from a textual source. It has long been acknowledged that textual information is a potentially useful, but largely untapped source of valuable knowledge in a number of domains, ranging from complex medical diagnosis to automobile engine repair. The problem at hand is to exploit those sources for use in knowledge-based systems. This is a considerably more ambitious task than most information extraction systems try to solve. Given the vast potential of textual sources, the applicability of information extraction must be at least considered for this research effort.

Research cited in Cowie and Lehnert [4] points to the cost-effectiveness of retrieving "shallow" knowledge. Shallow knowledge refers to knowledge that requires less effort for knowledge acquisition. It is also domain specific and *ad hoc* by nature. By using shallow knowledge, the data mining tool may be able gather large amounts of information in a short period of time. The idea is that each piece of knowledge taken from an external source comes at some measurable cost in terms of time, resources, effort, etc. Gathering large quantities of shallow knowledge may be more cost effective than painstakingly distilling a small amount of deeper knowledge. Also, the refinement of shallow knowledge will be simpler than the original knowledge extraction process.

According to Riloff and Lehnert [18], traditional information extraction normally consists of keyword searches based on statistical techniques. These techniques, while useful, have a number of limitations, some of which are

1. **Synonymy** - different words or phrases have the same meaning. For example, the words *make*, *manufacture*, and *produce* can all represent the same concept.
2. **Polysemy** - the same word can have multiple meanings in different contexts. For example, the word "fly" can mean conveyance aboard an aircraft or a small, annoying insect.

3. **Phrases** - some words are useful as indexes only when they are used in certain phrases. For example, the phrase "passed away" indicates that someone has died, but the two words used independently could have a different meaning (e.g. the phrase "the procession passed slowly away" does not suggest death).

4. **Local Context** - some words or phrases are useful as indexes only in certain contexts. For example, when mining for information about murders from terrorist attack, the terms "murder" or "killed" are not sufficient to indicate terrorism, but they do indicate terrorism in conjunction with words like "car bomb."

5. **Global Context** - some documents do not contain any words or phrases that are useful for indexing. The relevance of a document may depend on the context of any of its component parts. For example, the phrase "the man took the money and fled" clearly describe a robbery, even though none of the words alone would indicate a robbery.

The work done by Riloff and Lehnert [18] is aimed at using information extraction as a basis for text classification. Text classification is a process in which one or more categorical labels are associated with documents. While in-depth natural-language processing is an enormously difficult task, information extraction is a more tractable and robust technology. The goal of their work was to classify texts in a highly precise manner for categorization. The high precision aspect refers to the ability of their system to be as liberal or as conservative as the user desires in classifying the documents. Most systems of this type retrieve a small number of relevant documents, most of which are almost guaranteed to be useful. However, the approach used in Riloff and Lehnert [18] allows the user to make tradeoff decisions between relevancy and the amount of information extracted. Of course, once a document has been classified as relevant, the task of actually gathering the information still remains.

The goal of information extraction is to extract *specific types* of information from a document. The primary advantage of information extraction is that it allows significant amounts of irrelevant information to be disregarded. This restricts the search space and facilitates other subsequent data mining operations. The relevance of information extraction techniques to this research effort is discussed further in Section 3.1.

2.6.2 Extracting Belief Networks (Dependency Networks) from Databases

According to Sarkar and Murthy [21], a belief network is any network that stores probabilities using "probability calculus." They are directed, acyclic graphs where nodes represent propositions and arcs represent dependencies between the propositions. The belief assigned to the various propositions are stored as probabilities, and the strengths of the propositions across the network are represented as conditional probabilities. Under this definition, a BKB could be classified as a belief network. In a belief network, the representational flexibility of the network with a given connectivity is inversely proportional to the computational complexity necessary to make inferences across that network, so the knowledge engineer must always balance computability with representative strength. In Sarkar and Murthy [21], work is cited by Herskovitz and Cooper [3] that introduces a heuristic, K2, to extract the most probable belief network from a database. The suitability of K2 in solving our research problem is further discussed in Section 3.2.

2.6.3 Association Rules

Association rules are those in which one or more items in the antecedent of an implication are correlated with one or more items in the consequent with some level of confidence and support. An example of an association rule would be, "If a supermarket customer buys bread and eggs, he will also buy milk with a 90% probability." In this rule, the purchase of bread and eggs comprise the antecedent, the purchase of milk the consequent, and the value 90% is the confidence [1]. When searching for a rule of the form[6] X $\Rightarrow$ Y (read "X implies Y"), that rule has a confidence value of *C* if *C%* of the database records containing X also contain Y. If *S%* of the records in the database contain both X and Y, then the rule has a support value of *S* [2].

Association rules can have enormous implications in many domains. For instance, in retail sales, association rules can be employed in the areas of catalog design, add-on sales, store layout, and customer segmentation based on buying patterns [2]. Some examples are shown below [1]:

- *Find all rules that have "Diet Coke" as the consequent.* These rules may help retailers devise strategies to boost sales of Diet Coke.
- *Find all rules that have bagels in the antecedent.* These rules would indicate which products might be effected by discontinuing bagels.
- *Find rules relating shelf A in the store with shelf B.* These rules would help retailers to plan the layout of the two shelves by stocking items whose sales are related.

2.6.4 Exploiting Multiple, Heterogeneous Sources

With today's widespread use of database technology, there are a plethora of data representation formats. A potentially serious challenge to data mining operations in this unfriendly environment is the capability of interfacing with multiple, heterogeneous information sources. The developer of a data mining tool must be able to provide access to diverse and

[6] It is important to note that *causality* is not indicated in an implication because we have no way of inferring whether or not X causes Y to happen. To the contrary, we can only observe that when X occurs, Y also occurs some percentage of the time.

dynamic information sources. Some of the difficulties of working with such sources include the following [15]:

- **The exploitation of data that is unstructured or semi-structured, having no regular schema to describe it**. For example, free-form text may have some loose structure, but the "fields" may not have any explicit descriptions.
- **The dynamic nature of some environments**, where sources, their contents, and the meaning of their contents change frequently.
- **The access and integration of information that is intertwined**. Normally, a distinct integration phase is performed first, where data models and schemas are combined. Then the access phase is performed, where the actual data is extracted. In a heterogeneous environment, the interrelationships between pieces of information may not be apparent until a set of samples are examined. Those interrelationships may also change as further exploration takes place.
- **The need for human intervention in order to integrate or exploit the information**. On one extreme, information extraction may be done manually. At the other extreme, the system can perform the mining operation in a completely automated manner. However, it is usually necessary for a human to first study samples of the data and determine which procedures and methods to follow.

According to Papakonstantinou et. al. [15], the framework surrounding a heterogeneous information extraction tool should contain at least the following components:

- **Information Exchange.** The system components need to be able to exchange data objects freely. In order to accomplish this, there must be a standard data representation scheme and a format for transportation over a network. Once these standards are agreed upon, tools must be developed to translate between the various external representation formats and the standard, intermediate format.
- **Information Discovery and Browsing.** These tools will allow a human to become familiar with the semantics of objects, allowing the human to further direct the data mining process. These tools will enable the browser to query for sources of interest, request specific data from those sources, and to ask questions about the meaning of objects and their components.
- **Mediators.** A mediator is a component that collects information from a number of different sources, translates and processes that information, and makes the results available to other information sources.

Researchers at Stanford have devised a common language and information format for communication among heterogeneous entities in the form of a self-describing object exchange model [15]. In this model, called the Object Exchange Model (OEM), there is no notion of a fixed schema for an object or class of objects, because each object contains its own schema. The representation is generic enough to encompass all types of information, and each data object contains the semantic meaning necessary to describe that object [15].

Each object in the OEM model has the following structure:

Label	Type	Value	Object ID

The meaning of each of the four fields is as follows:

1. **Label:** A variable-length character string that describes what the object represents.
2. **Type:** The data type of the stored object. The types used in OEM consist of the standard atomic types **integer**, **string**, **real**, etc. as well as the aggregate type **set**.
3. **Value:** A variable-length value for the stored object.
4. **Object ID:** A unique, variable-length identifier for the object, or the special value *null*.

2.6.5 Autonomy vs. Flexibility

The ultimate purpose of any data mining or knowledge discovery system is to process large amounts of raw data, extract the most meaningful and interesting patterns, and present them to the user in a manner appropriate to the user's goals. Unfortunately, the evolution of the data mining field is such that a general-purpose, fully-automated system is far beyond reach. While considerable research has been performed on increasingly autonomous systems, user interaction remains an integral component of most systems. Very few knowledge discovery algorithms are totally autonomous, and those that are have limited applicability and generalizability. According to Matheus et. al. [12], autonomy and flexibility are inversely related. The more autonomous a system is, the less flexible it tends to be. More versatile systems have a wider range of discovery techniques at the expense of reliance on user guidance. Autonomy requires domain knowledge, while versatility implies domain independence.

2.6.6 General Difficulties in Data Mining

Little or no forethought is given to data mining during the design and creation of large databases. Databases are usually constructed with a particular organizational activity in mind. Consequently, a data mining application has to overcome numerous difficulties, each of which is representative of databases in general. Some of these difficulties include the following [12]:

- **The fundamentally dynamic nature of a database's contents**. The contents of most real-world databases are constantly changing. The designer of a real-time data mining application must use caution in order to prevent these constant changes from producing erroneous discoveries. This problem is sometimes overcome by focusing on snapshots of the data, and is appropriate whenever data storage activities take place in recurring, periodic intervals (e.g. yearly, quarterly, etc.). A possible drawback to this approach is the increased storage capacity needed to contain each individual snapshot.

- **The fact that erroneous data is replete within most databases**. Error-prone, manual collection of data is still commonplace. In fact, the purpose of some knowledge discovery systems is to identify data-entry errors. Also, useful patterns are seldom valid over more than a small percentage of the data. This requires the application of probabilistic techniques (e.g. it is very likely that someone with a high income is a good credit risk).

- **The incomplete nature of data**, either through missing record fields or from data missing from individual records. In relational databases, all records must have the same fields, even though some fields are blank in many records.

- **The wasted time and effort needed to process redundant data**. Duplicated information sometimes leads to the discovery of superfluous "facts," such as Profits = Sales - Expenses. This type of redundancy is called *functional dependency*. Another form of redundancy in data occurs when the value of one field is constrained by the value in another (e.g. Start Date ≤ End Date). To avoid the problem of discovering these relationships as "knowledge," the discovery system must know the database's structure and inherent, implicit dependencies ahead of time.

- **The sparse nature of data** in terms of the density of records over the potential instance space. For example, it is likely that few patients in a clinical database will have rare diseases.

- **The enormous size of many databases**. This problem has been the major driving force behind knowledge discovery efforts. Exhaustive searches through terabyte-sized databases are impractical, if not impossible. A knowledge discovery system must be able to focus on a set of fields or a subset of records, thereby limiting the size of the search space.

2.6.7 Idealized Model of a Data Mining System

The work done by Matheus et. al. [12] has led to a model for an idealized knowledge discovery system, which defines a system in terms of a collection of components. The components represent functional areas that should be present in a KDD system. Each component may not be explicitly identifiable, but the functionality represented that component should be present. The elements of this model are the following:

1. The **Controller**, which handles the invocation of the other components and parameter passing to each of them.

2. The **Database Interface**, which generates and processes database queries. Many knowledge discovery systems manipulate intermediate representations and do not interact with a database directly.

3. The **Knowledge Base**, which acts as a repository for domain-specific knowledge.

4. The **Focus**, which determines what portions of the data to analyze next.

5. The **Evaluation** element, which assesses the interestingness and usefulness of the extracted patterns.

In this model, autonomy comes from the controller, whose decisions are based on inputs from the user and on pre-existing domain knowledge. If the discovery task at hand is well defined, the controller may be able to perform in a more autonomous manner. However, in normal data mining operations, this is rarely the case. The user almost always has to assist the decision-making process in knowledge discovery.

Even though the database interface usually plays a subordinate role in data mining operations, it is nevertheless an important one. When dealing with very large databases, the interface becomes much more important because large databases cannot entirely fit into memory at once. This issue has been ignored in many data mining systems.

The focusing component of a data mining system determines which parts of the database should be examined. This component specifies which tables need to be accessed, which fields need to be returned, and how many records need to be retrieved. In order to perform efficiently and correctly, the focusing element must have detailed information about the database's structure and organization. It may use this information to limit the scope of the search to a relevant set of fields, but given the large size of many databases, the focus may also be forced to limit the number of records examined as well.

Databases contain many patterns, most of which are uninteresting and unusable. The *interestingness* of a pattern is the degree to which it is novel, accurate, and useful with respect to the user's needs. The evaluation component of a data mining tool determines the degree of interestingness for a particular pattern and decides which patterns to present to the user and in what order. Statistical significance is the usual means of determining interest, but raw statistics are not always indicative of interestingness. As an example, a small increase of sales in one region of the country might be more significant than a large increase in another region. The determination of interestingness will vary between applications, and even between databases. Consequently, significant domain knowledge must be applied when deciding interestingness [12].

2.7 Summary

In this chapter we present the context of this research and some of the essential issues involved therein. The impetus for this research stems both from knowledge acquisition and from knowledge base validation. We show a number of data mining techniques that could be used to solve incompleteness in knowledge bases. In Chapter 3, we consider each of these techniques in turn, show how each technique could be applied to our problem, and discuss the advantages and disadvantages to using each technique.

III. Methodologies Considered for Data Mining

Knowledge acquisition rarely ends when the knowledge base is assumed to be complete. Even the most detailed and careful knowledge acquisition activities induce inconsistency and/or incompleteness into the knowledge base. Verification and validation functions are normally sufficient to uncover most knowledge base problems, but once identified, incompleteness can be as difficult to address as the original knowledge acquisition task which created it.

Data mining techniques offer a number of possible solutions to the problem of incompleteness in knowledge bases. In Chapter 3, we consider a number of different data mining operations and show how they were considered for repairing incompleteness. We first present some methods we considered and discarded, such as information extraction and the derivation of belief networks from databases, and why we chose not to use them. In Section 3.3 we then illustrate the method chosen for this problem, mining association rules, and how we applied this technique to the problem of incompleteness. Some association rule issues particular to our research are also discussed as well as some specific techniques to make our data mining operations more tractable.

3.1 Information Extraction

One of the first data mining techniques we considered was that of information extraction. Given the tremendous amount of information stored in an easily retrievable text-based format, information extraction gives the promise of expansive sources for exploitation. Ignoring the vast numbers of existing on-line reference manuals, the Internet alone provides an enormous pool from which useful and interesting knowledge can be discovered. Unfortunately, of all the data mining techniques available, information extraction also proved to be the most difficult to implement.

Work done in Riloff and Lehnert [18] is used primarily for classifying texts with regard to their relevancy to a particular domain. The context of their research was to determine whether a given document discussed terrorist activity. Their system employs a part-of-speech dictionary with 5,436 definitions. This domain-specific dictionary alone took over 1,500 person-hours to develop by highly-trained researchers. Also, once a given text has been classified as relevant, it still has to be further processed, using subsequent information extraction or natural language processing techniques, to obtain the necessary information.

In solving our problem, we ultimately require the representation of information in probabilistic form. Even after a document has been classified as being relevant and an interesting set of facts or rules have been discovered, deducing a probabilistic value from raw text is no trivial matter. In fact, it is not always possible to deduce probabilistic values directly simply because they are not always explicitly represented in texts. While the possibility of a text-based system is intriguing, the amount of effort necessary to realized a fully functioning tool was beyond the scope of this research.

3.2 Belief Networks

As described in Chapter 2, algorithms have been proposed to extract the most probable belief networks from a given database [21]. This approach also seemed promising initially because the networks extracted by these algorithms had probabilistic representations. This meant that converting the information contained in one of these networks into a format suitable for BKBs would require minimal effort. However, beneath the surface, there were also difficulties associated with these methods. With one algorithm, K2, the number of different network structures that were possible grew exponentially with the number of variables in the database [3]. In addition, K2 employed a greedy algorithm that did not guarantee an optimal solution. Also, obtaining the best network configuration from a given database was NP-hard, therefore computationally prohibitive. Another algorithm, ITRULE, generated the K best rules from a database, where K was a user-specified parameter [21]. However, the algorithm did not yield the *best* set of K rules, and it did not guarantee that the generated network would contain *all* of the variables in the database.

Our requirement for repairing incompleteness necessitates the consideration (at minimum) of certain user-specified variables. One of our three fundamental forms of incompleteness (Category 1) requires establishing a relationship between two specific states (see Chapter 1). If such a relationship is needed between two component states, it is clearly unacceptable for a data mining solution to ignore either of them simply because they aren't part of the best K rules. Similarly, if a direct relationship exists within a given database between the states in question, the user shouldn't have to wait while an NP-complete algorithm derives an entire network. Given these limitations on the extraction of belief networks, we also discarded this approach and chose instead the formation of association rules to solve our problem.

3.3 Mining Association Rules

Of all the methods considered, association rules were the simplest to implement while also providing the best fit to our problem. The basic algorithm for determining an association rule is relatively straightforward, and the resulting rule is already in the correct form for incorporation into a BKB (or, for that matter, in any other probabilistic knowledge base). When searching for a rule of the form

$$X \Rightarrow Y$$

we search the entire database and compute the percentage of records containing X that also contain Y. This value, called the *confidence*, is treated in this work, as in other similar research ([2], [23]), as the *probabilistic strength* of the association rule. The intuition behind this is that if the rule $X \Rightarrow Y$ has a confidence value of C, then whenever we observe X we can expect to also find Y with a probability of C. From a statistical point of view, the confidence of a rule represents the correlation between its antecedent and its consequent.

The *support* for a rule is the value that represents the frequency of co-occurrence of all the variables in that rule within the database. For instance, in the above example, the support for the rule would be the percentage of records in the database that included both X and Y. In simple terms, the support represents the amount of "belief" or "faith" we have in the plausibility of the discovered rule. It is important that the support value for a rule is well chosen, for if the support is too low, unfounded rules with sufficient confidence values could be presented as valid associations. Consider once again the above example in a database containing 100,000 records. Suppose that the only occurrences of X and Y were together in the same record. The confidence for the derived rule would be 1.0 (or 100%), since *every* instance of X would be correlated with an instance of Y. However, the support, only 0.00001, would be insufficient in almost any imaginable circumstance for the formation of a rule.

A subtle but important point regarding confidence is that if A $\Rightarrow$ B with confidence *C*, it is almost certainly not the case that B $\Rightarrow$ A with the same confidence. (It is possible that both rules have the same confidence values given the right circumstances, but we cannot automatically draw this conclusion.) Consider the sample database shown in Table 1. If we're looking for a relationship between being married and the number of automobiles owned, then the rule would be:

Marital Status = Married $\Rightarrow$ Number of Automobiles = 2 with confidence 0.66

On the other hand, deriving the reciprocal relationship using the same states would yield the rule:

Number of Automobiles = 2 $\Rightarrow$ Marital Status = Married with confidence 1.00

Individual ID	Income	Marital Status	Number of Automobiles
100	35,000	Single	1
200	40,000	Married	2
300	30,000	Single	1
400	45,000	Married	1
500	46,000	Married	2

Table 3-1

3.3.1 Repairing Incompleteness With Association Rules

In a typical data mining application, the objective would be to generate all possible association rules whose support and confidence values exceeded some user-specified minimum. In this case, for each rule to be derived, the data mining tool would need to make as many passes over the data as the number of possible combinations of items in the antecedent. This number is exponentially large, and is often prohibitive for extremely massive databases. For our particular problem, it is sufficient to try to find associations between specific states (instances) in the database.

Referring back to the three primary categories of incompleteness listed in Chapter 1, categories 1 and 2 can be directly solved by a goal-directed search for association rules. By *goal-directed*, we mean the search for specific rules (i.e. a search for associations between specific states), rather than the search for all possible rules.

In incompleteness Category 1, we try to find an association rule of the form

$$X \Rightarrow Y$$

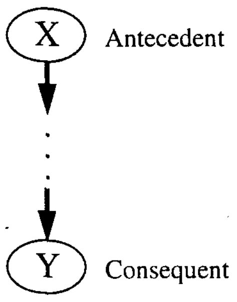
where X and Y are component states in need of some heretofore unspecified relationship (see Figure 3-1). If both the confidence and support values of the rule $X \Rightarrow Y$ meet the minimum values specified by the user, then the relationship is considered to be a direct one. On the other hand, if the minimum values aren't met, the data mining tool searches for all possible relationships of the form

$$X \Rightarrow Z_i,\ i = 1, 2, .. n$$

where n is the user-specified branching factor[7] and the confidence of the rule $X \Rightarrow Z_j$ is greater than or equal to the confidence of $X \Rightarrow Z_{j+1}$. The tool then tries to associate each Z_i with Y by finding rules of the form

$$Z_i \Rightarrow Y$$

If such associations are not possible, the process continues until either some associative relationship is found to Y, or until a user-specified lookahead value is exceeded. This lookahead value is used to specify the maximum number of intermediate relationships allowed between the original antecedent (X) and consequent (Y). For instance, a lookahead value of 1 would allow only a single intermediate state between X and Y (e.g. $X \Rightarrow Z \Rightarrow Y$). Note that it is feasible that each individual Z_i branch could lead to the consequent Y.



Category 1

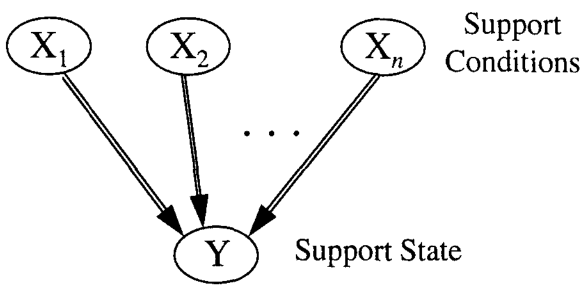
Figure 3-1

In Category 2, illustrated in Figure 3-2, we look for support conditions that are immediately related to a given node. In terms of data mining, this reduces to searching for all association rules of the form

$$X_i \Rightarrow Y, \text{i} = 1, 2, .. n$$

[7] The meaning of the term *branching factor* is used in this work to either specify the maximum number of intermediate associative relationships considered (Category 1) or the maximum number of supporting association rules that will be presented to the user (Category 2).

where Y is the state for which support is needed, *n* is the user-specified branching factor, and where the confidence of the rule $X_j \Rightarrow Y$ is greater than or equal to the confidence of $X_{j+1} \Rightarrow Y$.



Category 2

Figure 3-2

Finally, in Category 3, we are given a component (an attribute in database terms) for which new states are needed. The data mining tool searches the database and extracts all possible states for that component. This is a reasonably straightforward process for categorical (non-numerical) components. We simply examine each record and if we discover a previously unencountered state, we add the new state to a list. However, much more work is involved for numerical components.

3.4 Discovering Numeric States

There are several problems concerned with choosing states, or intervals, for numeric components. If the number of states for the component is large (i.e. if the size of each interval is small), then the support for any state will probably be low. As a result, any potential rules involving the numeric component may never be discovered. Conversely, there is always some information lost whenever we partition numeric values into intervals. This loss increases as the size of the interval increases. Some rules may only have sufficient confidence when the numeric interval consists of a single value. Referring back to the sample database in Table 1, the associative rule

$$\text{Number of Automobiles} = 2 \Rightarrow \text{Marital Status} = \text{Married}$$

has a confidence of 1.00, or 100%. However, when specifying a range in the antecedent,

$$\text{Number of Automobiles} = 1 - 2 \Rightarrow \text{Marital Status} = \text{Married},$$

the confidence is reduced to 0.60 [23]. Clearly, when dealing with numeric database attributes, support and confidence values are inversely related.

To alleviate this problem, we could consider all possible continuous intervals involving a specific numeric component. The problem of losing support disappears, because we can combine adjacent intervals or values in order to increase support when necessary. We may lose confidence, but we can still increase the number of intervals while still maintaining a minimum support. Unfortunately, there are problems with this approach also.

If a numeric component has n different values, there are, on average, n^2 different intervals that include each distinct value or interval. Computing the ideal set of interval ranges will drive up the execution time inordinately. On the other hand, if a value or interval of a numeric component has adequate support, then so will any interval containing it. Thus, the number of rules we derive based on that interval explodes. Many of these rules will be uninteresting. So once again, there is an inverse tradeoff between execution time and the number of derivable interesting rules [23]. To balance all of the aforementioned factors, we set the number of intervals, or states, for each numeric component to be $1/S$, where S is the user-specified support value (see Srikant and Agrawal [23] for a detailed proof).

3.5 Pruning the Search Space

In performing a Category 2 data mining operation, and possibly during a Category 1 operation (see Figure 3-1 and Figure), it is necessary to compare a component state with all the states of all the other components in the database. For instance, if there are A components in the database, and if each component has on average S states, then the tool would have to make $(A - 1) \times S$ passes through the database. If the database is quite large, as most modern databases of interest are, these passes can be prohibitive.

Our approach to solving this problem, at least in part, was to compute the support of each state during the initial formation of states. It should be obvious that, in order to determine the number of states for a component, and the extent of each state, a single pass will have to be made through the database for that component. As we make that pass, we compute the amount of support for each state. Recall that the support value is the number of occurrences for a given state divided by the number of records in the database.

It is also necessary to make a pass through the database each time we attempt to associate two different states with each other. In order to determine the correlation between two states, we must count the number of times they occur together in a record versus the total number of times they occur in the database. If at any point we consider comparing two different states, say X and Y, we can examine the minimum of their individual support values to determine whether it is even possible for their combined support to exceed the user-specified minimum. A more formal definition of support is

$$\textit{support}\,(X \Rightarrow Y) = | X \cap Y | \,/\, | \text{database} | \qquad (1)$$

Clearly, the greatest support value a rule can have is the minimum of the support values of the two states. If the user has directed the mining operation with a minimum support of 20%, and if we consider two states X and Y with support values of 10% and %3 respectively, then we know the most the support for the rule can be is %3, which does not exceed the minimum value. We can therefore reject the possibility that these two states could be related, thus saving an entire pass through the database to investigate the hypothesis. Note that the most support that any pair of states could contribute to a rule would only occur when the confidence value of their association was 1.0 (i.e. when the antecedent always occurred in the same record with the consequent).

Typically, the formation of association rules is a two-step process [2], [8]. First, all itemsets meeting the minimum support criteria are uncovered. Second, all possible combinations of items in each itemset are enumerated, where each combination places a single item in the consequent and all remaining items in the antecedent. If the confidence value of that combination is above the minimum, then a valid association rule is concluded. Our approach differs in that we know ahead of time which singular items we need to have associated. By screening the individual items on the basis of support, as mentioned above, we perform a step analogous to Step 1 in the traditional algorithms.

Of course, we never allow alternate states of the antecedent to be considered in a rule. This would be meaningless and might possibly allow the formation of a contradictory rule. For example, if we were mining for a rule of the form

$$A = 1 \Rightarrow B = 7$$

and if we considered other states of A, we might conclude a rule of the form

$$A = 1 \Rightarrow C = 2 \Rightarrow A = 5 \Rightarrow B = 7$$

which is clearly contradictory, since $A = 1$ can never imply the occurrence of $A = 5$.

Finally, in order to avoid cycles or redundant rules, we maintain a list of all comparisons made. Each time we consider correlating two states, we first ensure those states haven't already been compared. This prevents the formation of rules like

$$X \Rightarrow Y \Rightarrow Z \Rightarrow W \Rightarrow Y \Rightarrow Z \ldots$$

It also prevents inclusion of the same sequence of associations in multiple branches of the results tree. Observe that in Figure 3-3, the sub-rule $Z \Rightarrow W$ is present in two different associative paths leading to the consequent V. Clearly this is superfluous and should be avoided.

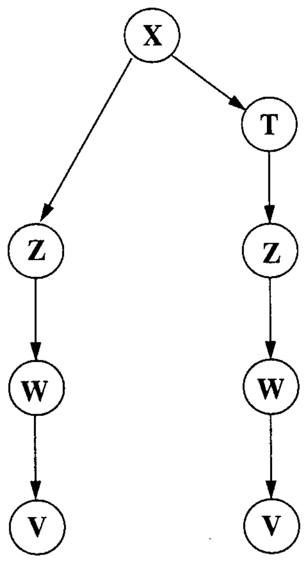


Figure 3-3

IV. Incorporating Data Mining Results

Extracting meaningful rules from raw data is only the first step of acquiring knowledge. The more important task of incorporating the results into the BKB which originally exhibited incompleteness remains to be accomplished. As stated in Chapter 1, the present technique for repairing incompleteness is the same as that of the original knowledge acquisition task. We simply poll the expert for the desired missing information and include it in the knowledge base. While knowledge acquisition of this type can be tedious, error-prone, and lengthy, automatically including new information obtained during data mining is also quite complicated.

4.1 Mutual Exclusion

Consider the knowledge acquisition task portrayed in Figure 4-1. The solid lines represent the state of a BKB prior to data mining, where a relationship between Z and X is missing, and the dashed lines represent the automatic inclusion of the rule $Z \Rightarrow X$ into the knowledge base, where $Z \Rightarrow X$ was discovered by the data mining tool[8]. In this example, by simply creating a direct link between Z and X, we have violated one of the eight constraints on the structure and consistency of a BKB, namely that of *mutual exclusion* [19]. The rule of mutual exclusion states that any S-nodes which support a common I-node must be mutually exclusive. In the BKB in Figure 4-1, it is possible for all of the support values for X (i.e. A, B, and Z) to be active at the same time.

[8] It is entirely possible that some subset of the data mining results already exist in the BKB. In that case, we simply augment the BKB with the remaining portion of the results.

There is nothing to differentiate between the two support conditions. This would mean that it would be possible for both of the S-nodes leading into X to be active simultaneously, which leads to a conflict, since only one S-node parent of a given I-node can be active at any given time.

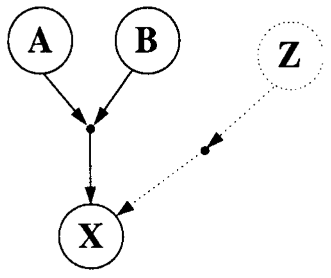


Figure 4-1

Figure 4-2 represents two possible solutions (depicted in dashed lines) to the mutual exclusion problem. If we assume that conditions A, B, and Z represent Boolean instantiations, then we can exclude one support condition by including the negation of the other support condition. In this example, we can either include the negation of Z in the A-B support, or we can include the negation of B and A in the Z support. If there were more supports for X, then each additional support would need to be included in this exclusion process. Conversely, if A, B, and Z were non-Boolean instantiations, then it would be necessary to enumerate all possible combinations of states for each of the supports of X. Applying this process over an extended period of time, we would cause the number of supports in the BKB to explode factorially, making future inferencing all but impossible.

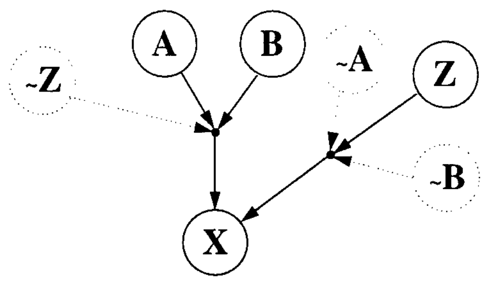


Figure 4-2

Our alternative to this clearly unacceptable method is to create a new form of node in the BKB called an *exclusion node*[9], or E-node (see Figure 4-3). An exclusion node, as its name implies, serves the purpose of excluding two or more supports of an I-node. To perform automated exclusion for the supports of a given I-node, in our example X, we create a new component whose name appropriately indicates that it is the exclusion of X (called E(X) in Figure 4-3). This new component will necessarily have as many states as X has supports. We reconnect the previous supports of X as supports of E(X), and connect the individual states of E(X) as the new supports of X. The probabilities of X's original support nodes, in this case P_1 and P_2, remain the same, thus preserving the intended semantics of the original BKB configuration. The support values leading out of each state of E(X) (in this case P_3 and P_4) are all set as high as possible and then normalized[10] with respect to all other states of X in order to alter the BKB semantics as little as possible [19]. Since only one state of a given component can be active at one time, only one of the exclusion nodes will ever be active. Since each of the new supports of X represents a different state of the same component, the supports of X are, by definition, mutually exclusive of each other. Hence the incompleteness originally identified is

[9] In implementation, each exclusion node is simply another I-node whose name indicates the fact that it excludes the supports of some other I-node.

[10] One of the probabilistic constraints of a BKB is that all supports for a given component sum to less than 1.0 (see [19]).

repaired, the integrity of the BKB is restored, and the BKB semantics intended by the user are preserved as accurately as possible.

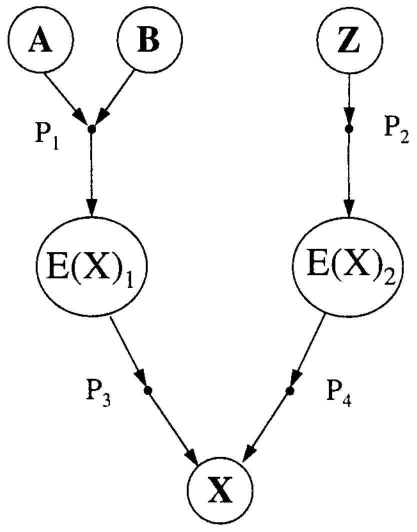


Figure 4-3

4.2 Advantages of this Method

On the positive side, E-nodes will indicate to the user where to go in order to rectify any occurrences of the mutual exclusion problem manually. If necessary, they can effect their own changes, based on their own expertise and domain understanding, while preserving the correctness, consistency, and intended meaning of the BKB. This should be easier for the expert to accomplish because of his knowledge of the problem domain's interrelationships. It will also be possible to remove the E-nodes at a later time, either manually or automatically, without effecting the semantics, structure, or consistency of the BKB, as long as the supports that originally violated the mutual exclusion property are addressed by the user. As another possibility, leaving the E-nodes in the BKB should require a minimal amount of storage, as compared to the remainder of the BKB.

4.3 Disadvantages of this Method

Unfortunately, there are drawbacks to this approach. The creation of E-nodes will eventually clutter the BKB if they are not dealt with by the user. This will put the burden of BKB maintenance on the user, such that he will be responsible for periodically examining the BKB and manually repairing any inconsistency indicated by the presence of an E-node. With this in mind, even though E-nodes leave the BKB in a consistent and correct state, their presence should be viewed as a temporary solution to an inconsistent configuration of the BKB which should be addressed as soon possible.

Attempts are currently being made to give the user of PESKI a visual representation of the BKB through the use of the DaVinci ™ graphical display tool, and the presence of numerous E-nodes may prove to be visually distracting. Their appearance may distort the semantic meaning of the BKB connections as perceived by the user (e.g. Phlegmatic Cough $\Rightarrow$ Exclusion(Tuberculosis) $\Rightarrow$ Tuberculosis), even though the actual inferential relationships of the domain are preserved intact.

The cluttered display problem could be solved by merely filtering out the E-nodes before presenting them to the user, but this would require extra work on the part of the user interface. This would entail displaying all the nodes and arcs around an E-node, effectively treating the E-node as though it were a single arc. This could also frustrate the user if he was attempting to locate and remove the E-nodes, so this filtering property of the display should probably be optional.

V. Implementing the PESKI Data Mining Tool

One of the key properties of PESKI is that it is a fully integrated suite of tools performing a wide range of tasks whose ultimate purpose is to create a customized knowledge-based system. Here we present a brief overview of the structure and functionality of PESKI. We then present the PESKI Data mining tool and show how this tool integrates with the other PESKI subsystems.

5.1 The PESKI Framework

At a conceptual level, PESKI consists of four major components [20]

1. **Natural Language Interface** - facilitates the processing of queries and the presentation of results in English.
2. **Inference Engine** - performs the reasoning in response to user queries and selects the reasoning strategy to apply.
3. **Explanation and Interpretation** - keeps track of the inferencing path taken to produce a solution.
4. **Knowledge Acquisition and Maintenance** - allows the user of PESKI to incorporate new or modified information into the knowledge base.

As mentioned in Chapter 2, PESKI is implemented as a combination of the following closely interconnected subsystems:

1. Intelligent Graphical User Interface
2. Inference Engine

3. Knowledge Acquisition
4. Verification and Validation
5. Data Mining

Figure 5-1 illustrates the interaction between the separate elements. The User Interface acts as a system controller and arbitrates the intercommunication between the separate entities. The Knowledge Acquisition module allows the user to create and modify a BKB. Verification and Validation ensures the BKB is consistent based on the user's test cases. The Inference Engine finds a solution in the BKB based on submitted evidence and provides that solution to the user. The broken arrows represent future connections between the Data Mining Tool and other modules (see Chapter 6 for a comprehensive discussion on future implementation).

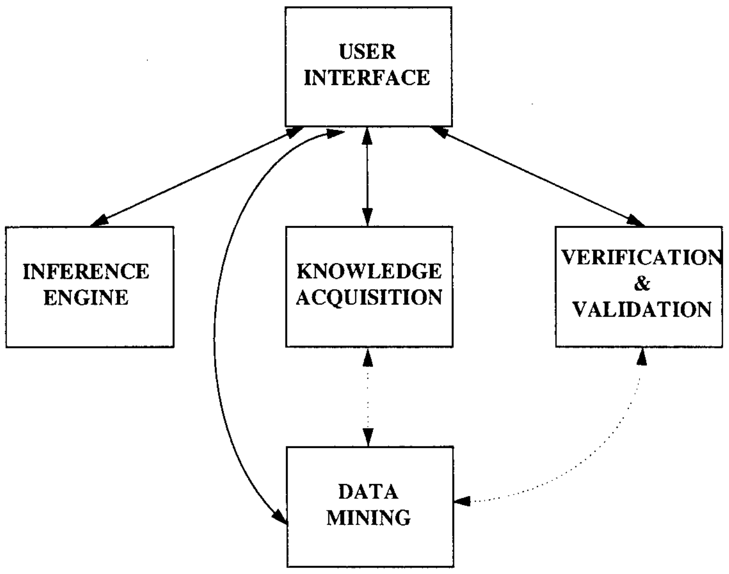


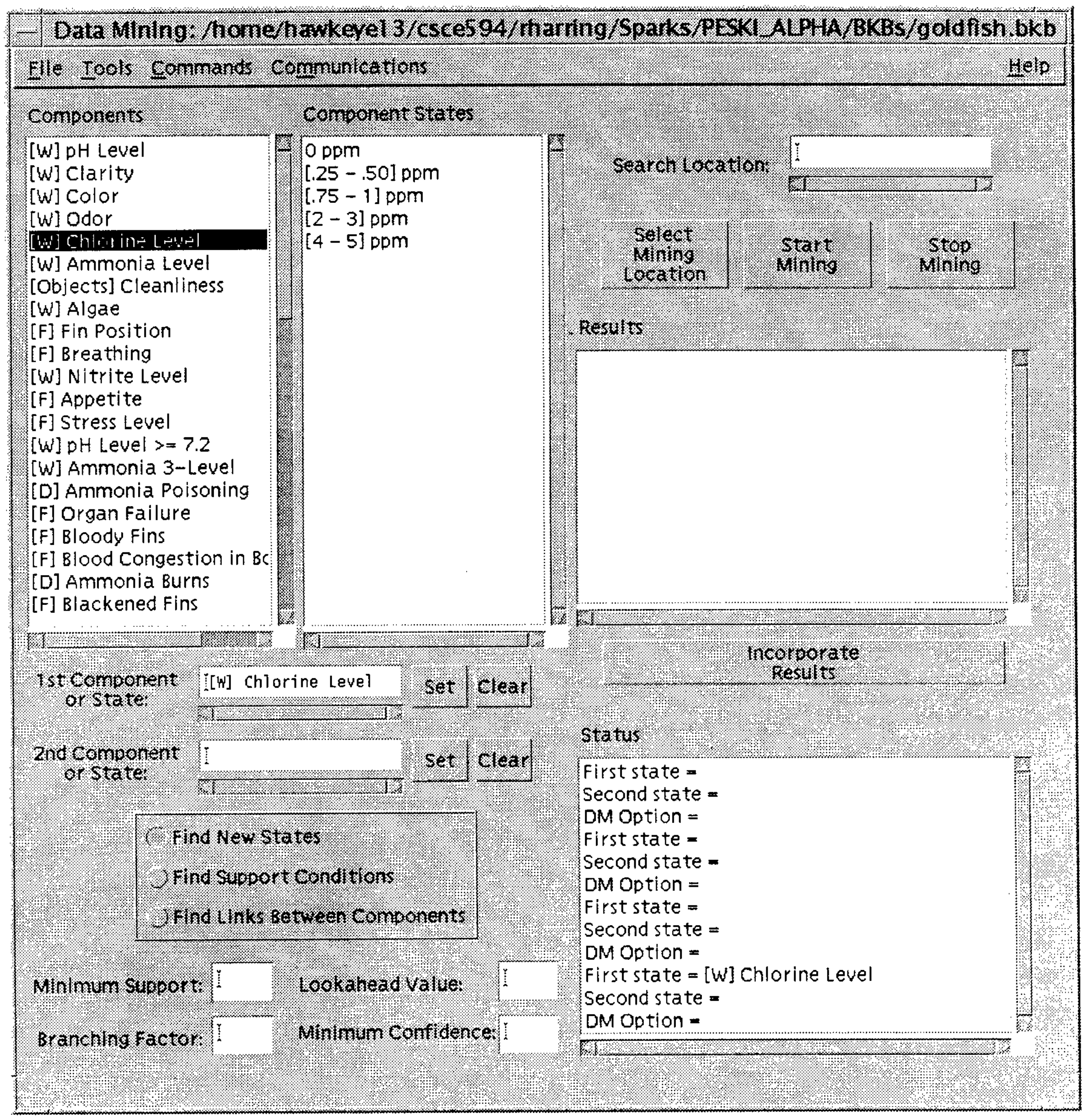
Figure 5-1

5.2 Data Mining and the Big Picture

By its very nature, the data mining tool is categorically subordinate to knowledge acquisition, since data mining is a knowledge acquisition activity. Data mining simply takes place after the initial knowledge acquisition is assumed to be complete. On a functional level, however, data mining falls within the purview of verification and validation, since it is this module that uncovers the incompleteness necessitating data mining. In its present state, the verification and validation module reports incompleteness to the user, who then invokes data mining manually. In the future, this invocation should take place automatically during verification and validation whenever possible. The user should only be notified after a mining operation has taken place and modified the BKB so that he can approve or disapprove the knowledge base modifications.

5.3 Data Mining Operations within PESKI

Consider PESKI's Data Mining control screen shown in Figure 5-2. In this example, the goldfish diagnosis BKB *GF2* has already been loaded into PESKI (see [11] for details about *GF2*). One of the existing components, *Chlorine Level*, has been selected and the *Find States* mining operation is about to take place. None of the required numeric parameters (*Minimum Support*, etc.) have been set. The *Status* window, in the lower right-hand corner, is used to communicate interim messages to the user of the data mining tool. In this case, the user has been notified that *Chlorine Level* has been selected for this mining operation. In the upper right-hand corner of the screen, the user indicates the mining source in the *Search Location* window. This can be done either manually or by browsing the contents of any available disk drive. The *Results* window displays the outcome of the data mining operation in a textual format. The user is able to select these results individually for incorporation into the BKB. Note that the user has the ability to terminate any data mining operation by pressing the *Stop Mining* button in the upper right-hand corner.



Components:
[W] pH Level
[W] Clarity
[W] Color
[W] Odor
[W] Chlorine Level
[W] Ammonia Level
[Objects] Cleanliness
[W] Algae
[F] Fin Position
[F] Breathing
[W] Nitrite Level
[F] Appetite
[F] Stress Level
[W] pH Level >= 7.2
[W] Ammonia 3-Level
[D] Ammonia Poisoning
[F] Organ Failure
[F] Bloody Fins
[F] Blood Congestion in Bc
[D] Ammonia Burns
[F] Blackened Fins

Component States:
0 ppm
[.25 - .50] ppm
[.75 - 1] ppm
[2 - 3] ppm
[4 - 5] ppm

1st Component or State: [W] Chlorine Level

2nd Component or State:

Status:
```
First state =
Second state =
DM Option =
First state =
Second state =
DM Option =
First state =
Second state =
DM Option =
First state = [W] Chlorine Level
Second state =
DM Option =
```

Figure 5-2

VI. Future Work

Any research project worth considering rarely solves a given problem in its entirety, and this work is no exception. Here we present some ideas for the future researcher of data mining techniques. We discuss some areas that will improve the performance of our data mining tool as well as some alternate applications for the tool that were not part of our original design. We also suggest some extensions to the tool that will enable it to perform its designated task more effectively.

6.1 Selecting Alternate Data Mines

Of course, when constructing the ideal data mining tool, there are numerous issues that need to be addressed. In order to function successfully, a tool must be able to maneuver through various databases, each potentially having a different format. The tool must accept these differing formats effortlessly and, as presented in Chapter 2, should be able to represent all pieces of data in a single, universal intermediate format. This intermediate format must be flexible as well as accurate so that it doesn't encumber the otherwise difficult task of data mining.

Whenever a mining operation cannot be successfully completed, the model tool should be able to apply a number of different strategies to further the mining effort. Normally, the user is responsible for selecting the data mining source. However, in a perfect situation, the system would autonomously identify and prioritize a set of possibly heterogeneous sources and select its own data mine or mines. This could be done using embedded heuristic rules (e. g. database Format 1 is always better than Format 2, or newswire text is preferable to html's). Alternatively, the data mining tool could also become truly intelligent and "learn," through trial-and-error experience, which sources were more applicable to the problem domain and which evaluation criteria were the better predictors of usefulness. The tool should be able to contemplate new sources for exploitation, classify them based on their significance and relevance, and select the most promising ones to exploit. This would entail examining each database and classifying it with regard to its relevance to the domain of interest. From the user's perspective, one would hope that the tool would not indiscriminately search every available database. Contrarily, the tool should *consider* each database as being a potential source of knowledge, but it should only mine the most promising ones for the desired information. In short, a good data mining tool should adapt to the difficulties it encounters and overcome them without supervision or user intervention.

This raises another issue, namely that of which *metrics* to use in order to differentiate between potential sources. The tool must decide how to prioritize its sources so that it can search the most promising ones first. There are many factors which could influence this decision, some of which include the following:

- **Ease of access** - the tool may want to examine databases or files on a local drive before looking at files across a network or on the Internet.
- **Ease of exploitation** - by virtue of query optimization, an object-oriented or relational database might be easier to mine than a flat file.
- **Promise of interestingness** - in the medical domain, we might want to examine a mining source that contains 500 disease references before we consider one with only 200.

- **Estimated accuracy anticipated from a result** - medical journals contain more accurate information than newspaper articles, though both could be valid sources.
- **The type of data mining problem being considered** - when researching terrorist activity, the newswire might be the most valuable source of information because of its timeliness, even though it could be more difficult to exploit than a more structured representation such as a relational database.

It would be useful to express each of the above measures in some weighted numerical fashion, like a percentage or a probability of usefulness. Weighting each of the measures would allow the tool to place varying amounts of importance on different factors based on the situation and type of mining operation.

As an aside, the utilization of multiple sources for data mining can also be used to increase the validity of the extracted information. Even though more effort is involved, there are a number of benefits to mining from several databases, namely [17]:

1. It maximizes the probability of extracting all the core associations for a particular problem domain.
2. During knowledge acquisition, it forces the knowledge engineer to consider different perspectives on rules.
3. It allows for the fact that certain cases may not be fully covered in any single source.

6.2 Using Natural Language Processing to Guide Data Mining

Quite often, a data mining operation aimed at a particular database is capable of providing meaningful, interesting, and useful information. However, it is frequently the case that either the desired relationships aren't contained in a given database, or that the necessary concepts are disguised or hidden from the data mining tool. In these situations, it becomes necessary for the tool to "spread out" its search context and to investigate other ancillary means of discovering information. It should be able to consider secondary definitions and synonyms for each of the attributes currently being considered. This ensures that a concept will not be overlooked simply because of incorrect phrasing or inadequately descriptive labeling.

One possibility for the data mining tool would be to re-examine each of the fields to determine whether the needed attributes were represented under a different label or at a different level of abstraction. This would require the tool to be able to reformulate or rephrase queries based on information synonymous to the original query. It would also be helpful if the database itself stored a set of related concepts describing each attribute of each relation.

It would be beneficial to invoke some form of a natural language processing or information extraction function to draw out the relevant concepts from the database's meta-data. The tool would have to compare a list of terms equivalent to the query against each of the database attributes' synonyms to see if and where any matches occurred. It could also exploit hierarchical relationships within terms (e.g. *is-a* relations such as Tuberculosis *is-a* Lung Disease). The tool could examine the header of each database relation, in addition to any descriptive comments contained about each attribute, and compute a relevancy metric in order to determine whether any of the fields were appropriate for the mining operation.

One important precaution to take in this situation would be to avoid over-generalizing the query terms. For instance, if the original data mining query was between the instances Cough = Phlegmatic and Disease = Tuberculosis, the tool might consider searching for a relationship between Symptom and Disease = Tuberculosis, or between Cough = Phlegmatic and Lung

Disease. However, a potentially undesirable mistake would be to search between Symptom and Lung Disease due to the highly indeterminate nature of the query.

6.3 Alternate Uses for the Data Mining Tool

In addition to performing knowledge acquisition during validation to repair incompleteness, we recognize that the data mining tool could increase our confidence in the validity of a given test case used during validation. The data mining tool can provide a "sanity check" of each test case by determining whether a relationship exists between each element of the evidence set and the elements of the solution set. By submitting a test case, we assert that the answer should somehow be concluded from the evidence. If this is true, then we should be able to establish a probabilistic relationship between the evidence and the answer through data mining. After all, if the evidence can "cause" the answers to occur, then we should be able to find at least an indirect relationship between the evidence and the answers using data mining results. With this said, whenever a test case fails, it may mean that the test case, not the BKB, is invalid. So whenever a test case fails to yield the expected results, we can subsequently submit a query or a set of queries to the data miner prior to reinforcement learning, in order to substantiate the legitimacy of the test case.

Similarly, the data miner can perform the same kind of validation on the BKB itself, both during and after knowledge acquisition. We create test cases to validate a BKB, but each test case is based on the structure of the BKB itself, as well as on our understanding of the problem domain. Since no expert is infallible, there is a distinct possibility that the BKB structure elicited from that expert could be also in error. We could compare our BKB configuration to a set of data mining results to cross-check the structure, or we could consult the data mining tool to give an alternate, and possibly more efficient, structure to consider. Also, according to Sarkar and Murthy [21], in a problem domain where numerous components exist, the probabilistic values taken from an expert are more likely to be in error than those derived from an existing database. With this in mind, it may be reasonable to either authenticate the probabilities given by the expert or to simply derive the probabilities through data mining whenever possible[11].

Finally, and perhaps most obviously, the data mining tool can be used during initial knowledge acquisition to establish the nuclear set of rules to be placed in the BKB. Once the knowledge engineer knows the central elements and concepts of the problem domain, he can invoke the data mining tool to establish the fundamental relationships between those concepts. By doing this, the knowledge engineer could greatly reduce the amount of time required during interviews with the domain expert.

6.4 Extending the Functionality of the Data Mining Tool

In addition to the functionalities listed previously, there are a number of extensions that can be made to the data mining tool to make it more versatile. The original motivation for the tool was to find relationships between an antecedent and a consequent, where each side of the implication was a singular entity. However, real-world relationships are often more complex. Rules in a BKB frequently involve multiple terms logically ANDed together in the antecedent. The data mining tool should be extended to allow for this case by searching for relationships

[11] It is important to note that the assignment of exact probabilities is not crucial to the overall success of inferencing over a BKB (see [1]). The probabilistic values are refined through inferencing and validation. However, experts don't always reason in terms of probabilities, and the use of probabilistic values may be awkward and unnatural for them.

between sets of items, as opposed to single items, during rule formation. Agrawal and Srikant [2] present one algorithm for finding association rules between itemsets. Essentially, their algorithm begins with all itemsets of size 1, and extends the size of each set until no larger sized sets exist that meet the minimum support conditions. They then apply techniques similar to ours to mine for association rules between the large itemsets. Houtsma and Swami [8] also express a number of algorithms dealing with itemsets that can be expressed in the form of SQL queries.

There are several other areas in this realm which need to be addressed. Some of these areas include the following:

- *Automatic invocation of the data mining tool by PESKI's Verification and Validation sub-system.* At present, the tool must be invoked manually by the user of PESKI, but since incompleteness is uncovered during knowledge base validation, it would be possible to begin data mining to fill some incomplete area without the user's involvement or knowledge. In the situation where a suite of test cases are being processed, the test case exposing the incompleteness could be skipped while the next test case is considered.

- *Natural Language Processing (NLP) exploitation of text documents for probabilistic information.* While this capability is probably the least likely to extract probabilistic rules, it is entirely possible that there may be no databases available, relational or otherwise, that pertain to the problem domain. In this case, text-based information may be the only source accessible for exploitation.

- *Formation of dependency networks from databases for initial knowledge acquisition.* This technique, mentioned in Section 2.6.2, is not suited for addressing incompleteness due to its computational complexity. However, it may prove useful during the early stages of knowledge base creation by inferring an initial knowledge base structure based on existing database dependencies.

6.5 Conclusions

This research develops a methodology and a tool for mining association rules and incorporating those rules into a knowledge base. The results extracted by our tool are intended for incorporation into a BKB, but are in a form suitable for incorporation into any knowledge base with a probabilistic representation. The tool extracts information in one of three forms depending on the type of incompleteness encountered during knowledge base validation. These three forms are as follows:

1. A series of association rules relating an antecedent state to a consequent state.
2. A set of associations related to a single consequent.
3. A set of states for a particular component in the database.

Our data mining tool is designed specifically for integration into PESKI, though the techniques we present are general enough for incorporation into other comparable systems.

Most data mining tools attempt to derive *all possible* rules meeting minimum support and confidence criteria. The enormous size of most real-world databases, and the corresponding overabundance of rules they contain, restrict the feasibility of data mining and effective comprise a "bottleneck" to database mining operations. Our approach is slightly different in that each mining operation is aimed at finding specific rules relating two or more database attributes. We call these focused operations *goal-directed data mining*, the goal being the association of two specific states to each other, either directly or indirectly. We enhance the effectiveness of data mining operations by:

1. Always attempting to find an association rule involving a particular state.
2. Eliminating attempts at rule formation whenever possible by considering the support value of each state involved.
3. Preventing the same states from being compared more than once, thus avoiding circular or repetitive rules.

This work has presented some key issues regarding database mining activities with respect to knowledge acquisition with an emphasis on probabilistic knowledge base validation. As both techniques and technology continue to mature, data mining will almost certainly play roles of ever increasing importance in both the domains of knowledge acquisition and verification and validation.

Bibliography

1. Agrawal, Rakesh, et. al. "Database Mining: A Performance Perspective." December 1993 *IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6.* 914 - 925.

2. Agrawal, Rakesh and Ramakrishnan Srikant "Fast Algorithms for Mining Association Rules." September 1994 *Proceedings of the 20th Very Large Data Bases Conference*, Santiago, Chile. 487 - 499.

3. Cooper, Gregory F. and Edward Herskovits "A Bayesian Method for Constructing Bayesian Belief Networks from Databases." 1991 *Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence*, Los Angeles, California. 86 - 94.

4. Cowie, Jim and Wendy Lehnert "Information Extraction." January 1996 *Communications of the ACM, Vol. 39, No. 1.* 80 - 91.

5. Frawley, William J. et. al. "Knowledge Discovery in Databases: An Overview." Fall 1992 *AI Magazine*, 57 - 70.

6. Gleason, Howard T. "Probabilistic Knowledge Base Validation." MS Thesis AFIT/GCS/ENG/95D-04. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December, 1995.

7. Gonzalez, Avelino J. and Douglas D. Dankel *The Engineering of Knowledge-Based Systems, Theory and Practice*, Prentice Hall, Englewood Cliffs, New Jersey.

8. Houtsma, Maurice and Arun Swami "Set-Oriented Mining for Association Rules in Relational Databases" *Proceedings of the International Conference on Data Engineering*, Taipei, Taiwan, March 1995. 25 - 33.

9. G. Kabanda "An Automatic Knowledge Acquisition Methodology for Agro-Meterological Data Analysis and Interpretation" *Applications of Artificial Intelligence in Engineering, 9th International Conference 1994* Central Computing Services, Causeway, Harare, Zimbabwe. 438 - 451

10. Kloesgen, Willi and Jan Zytkow "Machine Discovery Terminology." Internet document found at address "http://info.gte.com/~kdd/kdd-terms.html" (date unknown).

11. Lyle, Louise J. "A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction." MS Thesis AFIT/GCS/ENG/96D-17. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December, 1996.

12. Matheus, Christopher J., et. al. "Systems for Knowledge Discovery in Databases." December 1993 *IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6.* 903 - 913.

13. Moulet, M. And Y. Kodtratoff "From Machine Learning Towards Knowledge Discovery in Databases." 1995 *IEE Colloquium on Knowledge Discovery in Databases.* 5/1 - 5/3.

14. O'Keefe, Robert M., et. al. "Validating Expert System Performance." Winter 1987 *IEEE Expert*. 81 - 89.

15. Papakonstantinou, Yannis, et. al. "Object Exchange Across Heterogeneous Information Sources." *Proceedings of the 1995 IEEE 11th International Conference on Data Engineering*. 251 - 253.

16. Piatetsky-Shapiro, Gregory, et. al. "KDD-93: Progress and Challenges in Knowledge Discovery in Databases." *AI Magazine, vol. 15*, Fall 1994. 77 - 82.

17. Ram, Sudha and Sundaresan Ram "Design and Validation of a Knowledge-Based System for Screening Product Innovations." March 1996 *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 26, No. 2*, 213 - 221.

18. Riloff, Ellen and Wendy Lehnert "Information Extraction as a Basis for High-Precision Text Classification." July 1994 *ACM Transaction on Information Systems, Vol. 12, No. 3*. 296 - 333.

19. Santos, Eugene Jr. And Darwin O. Banks "Acquiring Consistent Knowledge in the Face of Uncertainty." *IEEE Transactions on Knowledge and Data Engineering* (1995) (Submitted to).

20. Santos, Eugene Jr. "A Fully Integrated Probabilistic Framework for Expert Systems Development." November 12, 1993 Research proposal from Air Force Institute of Technology to Air Force Office of Scientific Research.

21. Sarkar, Sumit and Ishwar Murthy "Constructing Efficient Belief Network Structures with Expert Provided Information." February 1996 *IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 1*. 134 - 143.

22. Shortland, Richard and Richard Scarfe "Digging for Gold." September 1995 *IEEE Review*. 213 - 217.

23. Srikant, Ramakrishnan and Rakesh Agrawal "Mining Quantitative Association Rules in Large Relational Databases" June 1996 *Proceedings of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada (to appear in).

Vita

Daniel Stein [redacted]. He was awarded dual Bachelors of Science in Mathematics and Computer Science from the University of Denver in 1990. Upon his commission, he was trained as a Communications-Computer Officer, and assigned to the (then) Strategic Operations Division, Headquarters Strategic Air Command. He arrived at the Air Force Institute of Technology in May 1994.

Permanent Address: [redacted]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1996	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE UTILIZING DATA AND KNOWLEDGE MINING FOR PROBABILISTIC KNOWLEDGE BASES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Daniel J. Stein III, Captain, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2750 P Street WPAFB, OH 45433-7126	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCS/ENG/96D-25
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Abraham Waksman 110 Duncan Ave. Bolling AFB, D.C. 20332	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Problems can arise whenever inferencing is attempted on a knowledge base that is incomplete. Our work shows that data mining techniques can be applied to fill in incomplete areas in Bayesian Knowledge Bases (BKBs), as well as in other knowledge-based systems utilizing probabilistic representations. The problem of inconsistency in BKBs has been addressed in previous work, where reinforcement learning techniques from neural networks were applied. However, the issue of automatically solving incompleteness in BKBs has yet to be addressed. Presently, incompleteness in BKBs is repaired through the application of traditional knowledge acquisition techniques. We show how association rules can be extracted from databases in order to replace excluded information and express missing relationships. A methodology for incorporating those results while maintaining a consistent knowledge base is also included.

14. SUBJECT TERMS Data Mining, Knowledge Discovery in Databases, KDD, Knowledge Acquisition, Bayesian Networks, Bayesian Knowledge Bases, Expert Systems.	**15. NUMBER OF PAGES** 65
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C -	Contract	PR -	Project
G -	Grant	TA -	Task
PE -	Program Element	WU -	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.