Discovering Multidimensional Patterns in Large Datasets by Knowledge Scouts

Ryszard S. Michalski
Kenneth A. Kaufman

MLI 99-7

DISCOVERING MULTIDIMENSIONAL PATTERNS IN LARGE DATASETS USING KNOWLEDGE SCOUTS

Ryszard S. Michalski[*] and Kenneth A. Kaufman

Machine Learning and Inference Laboratory
George Mason University
Fairfax, VA 22030-4444

MLI 99-7
P99-8

June 1999

[*] Also Institute of Computer Science, Polish Academy of Sciences

DISCOVERING MULTIDIMENSIONAL PATTERNS IN LARGE DATASETS USING KNOWLEDGE SCOUTS

Abstract

This paper presents the concept of a *knowledge scout*, an intelligent agent that operates within an *inductive database* to automatically search for target knowledge. A knowledge scout is defined by a script in *knowledge generation language* KGL-1, a high-level query language that integrates various data mining and machine learning programs with standard data and knowledge management operations in the inductive database (a system that integrates a database with inductive inference capabilities). In searching for target knowledge (e.g., strong patterns in data, or specific knowledge required by a user), a knowledge scout is guided by a model of the user's interests. Discovered patterns are represented in two forms, association rules in the *attributional calculus* (a description language with an expressive power between propositional and predicate calculus), and *association graphs*, which graphically represent relations expressed by the rules. The association graphs can depict simply and understandably multi-argument relationships among different concepts, with an indication of the relative strength of each interdependency, as measured by confidence parameters in the rules. Presented ideas are illustrated by two experimental knowledge scouts, one that seeks relations among lifestyles, environmental conditions, symptoms and diseases in a large medical database, and another that searches for patterns of children's behavior in the National Youth Survey database. The preliminary results indicate a high potential utility of the presented methodology for many data mining applications.

Keywords: Data Mining, Knowledge Discovery, Knowledge Scouts, Inductive Databases, Knowledge Visualization, Knowledge Generation Language, Association Graphs, Attributional Calculus

Acknowledgments

This research was done in the Machine Learning and Inference Laboratory of George Mason University. The Laboratory's research activities have been supported in part by the National Science Foundation under Grants No. IIS-9904078 and IRI-9510644, in part by the Defense Advanced Research Projects Agency under Grant No. F49620-95-1-0462 administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under Grant No. N00014-91-J-1351.

1 INTRODUCTION

When applying data mining tools to a large database, a user often has to conduct many trials and to backtrack various operations before a desired pattern is found. This process can be quite time-consuming, and makes it difficult to analyze large databases in which many different patterns can be revealed. Another difficulty concerns the problem of specifying the *target knowledge*, that is, the type of patterns that are likely to be of interest to a user. Obviously, one cannot define such patterns too precisely, as the whole purpose of the search is to find something new and unexpected. In addition, the target knowledge may change over time, as it depends on the user's prior knowledge and current goals. This means that a mechanism is needed for acquiring and monitoring the profile of the user's interests, and applying it in the search for target knowledge.

To address the problems mentioned above, the idea of a *knowledge scout* is proposed. A knowledge scout is an intelligent agent that automatically applies various data mining operators in search of target knowledge in a large database, which can be local or distributed. A knowledge scout operates within an *inductive database*, which is an integration of a conventional database with inductive inference capabilities. Therefore, before we present a method for constructing knowledge scouts, let us briefly discuss the concept of an inductive database.

In contrast to a conventional database, an inductive database can answer queries that require the synthesis of *plausible knowledge*, that is, knowledge that is not directly or deductively obtainable from the database, but can be hypothesized through inductive inference or other forms of uncertain inference. Such knowledge may be in the form of hypotheses about future datapoints, likely consequences from the data, generalized data summaries, emerging global patterns, exceptions from hypothesized patterns, suspected errors and implied inconsistencies, hypothetical plans synthesized from the data, etc. (Michalski, 1999). A general diagram of an inductive database is presented in Figure 1.

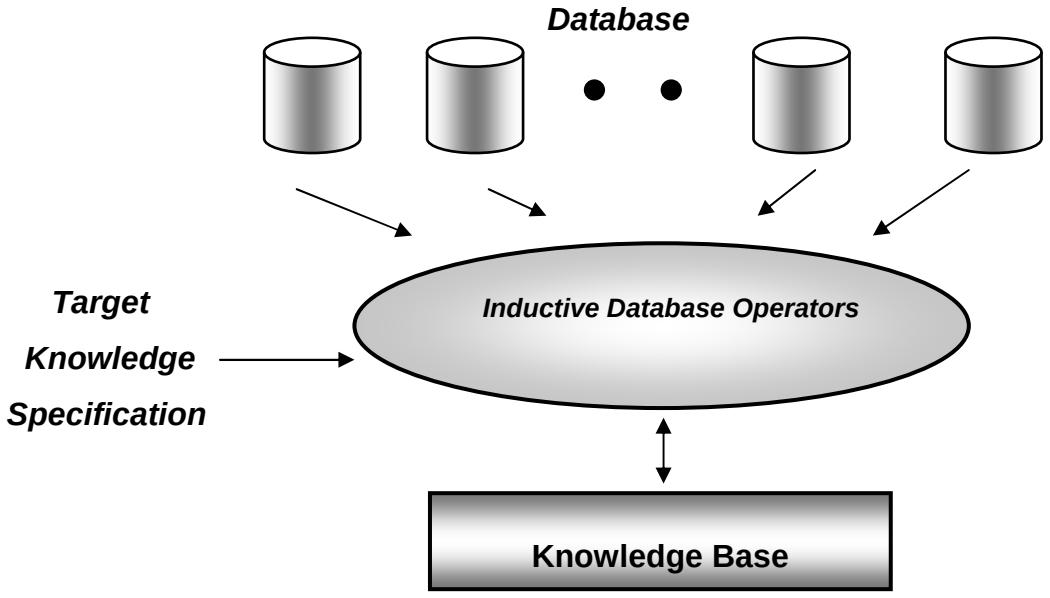

Figure 1. A general diagram of an inductive database.

The target knowledge for a scout is defined abstractly by specifying properties of pieces of knowledge that are likely to be of interest to the given user (or a specified group of users). An example of target knowledge can be data descriptions or patterns that have the highest quality according to some description quality measure (e.g., Kaufman and Michalski, 1999).

In order to synthesize target knowledge, a knowledge scout may execute long sequences of different kinds of operations involving data, intermediate results, and background knowledge. The latter is contained in the Knowledge Base, and may include domain constraints, the user's interests, and other relevant past knowledge. At every step of this process, an application of one operator may depend on the results of previous operators. The user's interests and past relevant knowledge are partially defined a priori, and partially constructed and updated during the scout's lifetime. To implement such capabilities, one needs to define one or more knowledge representation systems for expressing synthesized knowledge, and a knowledge generation language for defining knowledge scouts and target knowledge (or target patterns). The following three sections address these issues. Section 2 describes a knowledge representation system based on attributional rules, Section 3 describes a representation system based on association graphs, and Section 4 describes KGL-1, our initial knowledge generation language.

2 ATTRIBUTIONAL RULESETS

The language in which patterns or knowledge of interest are to be expressed is essential to the ability to discover them. If the language is too restricted, patterns will have complex expressions, and this in turn will make their discovery difficult. If the language is too rich, the pattern search space may become computationally prohibitive. In addition, an important practical criterion is that patterns should be easy to understand and interpret. Guided by such considerations, we employ *attributional calculus rules* for expressing patterns or knowledge of interest (Michalski, 1999).

The attributional calculus is an extension of propositional calculus in which literals (propositions and their negations) are replaced by *attributional conditions*. Such conditions represent relational statements that bind one or more attributes with a set of their values or other attributes. Each attribute has a domain and a type, the former defining its set of legal values, and the latter characterizing an ordering relationship among the values. Attributional calculus is based on variable-valued logic system VL1 (Michalski, 1975).

An attributional condition is in the form: **[L rel R]**, where **L** (*left side*) is an attribute, or one or more attributes with the same domain, joined by "&" or "v" (these links are called *internal conjunction* and *disjunction*, respectively); **R** (*right side*) is a value or a list of values joined by the symbol "v" or the word "or" (called *internal disjunction*), a pair of values joined by ".." (called *range*), or an attribute with the same domain as the attribute(s) in **L**; and

rel is a relational symbol from the set {=, ≠, >, ≥, <, ≤}. A condition **[L rel R]** is *true* (or *satisfied*), if expression **L** is in relation **rel** to **R**. For illustration, the following are examples and explanations of attributional conditions. Note that attributional conditions are easy to interpret and translate directly to corresponding natural language expressions.

[blood-pressure = normal]	*(the blood pressure of the patient is normal)*
[income = 50K..80K]	*(the income is between 50K and 80K)*
[color = red v blue]	*(the color is red or blue)*
[width & length > depth]	*(the width and length are both greater than the depth)*

Attributional rules used in this study are in the form **<decision> if <conditions**>, where **<decision>** is a single attributional condition, and **<conditions>** is a conjunction of one or more attributional conditions. These rules are a special case of the *parameterized association rules* (PARs), described in (Michalski, 1989). The association rules presented in (Agrawal, Imielinski and Swami, 1993) could be viewed as a specialized form of PARs.

Such attributional rules can be learned from a set of training examples using an operator based on the AQ-18 rule learning program (Kaufman and Michalski, 1999). The learned rules and their constituent conditions are output with annotations characterizing their importance, such as *support*, *disparity*, *completeness* and *consistency* for each condition in the rule and for the rule as a whole (the if-part). The support, denoted by *p*, is defined as the number of positive training examples that satisfy the given condition(s). The disparity, denoted by *n*, is defined as the number of negative training examples that satisfy the given condition(s). The completeness, denoted *compl*, is defined as p / P, where P is the total number of training examples in the positive class. The consistency, denoted *cons*, is defined as $(p / p + n)$. The program also generates other annotations, such as exceptions, ambiguity, rule quality, which are described elsewhere (e.g., Kaufman and Michalski, 1999).

The following example illustrates one of the attributional rules generated by a knowledge scout seeking demographic patterns in a World Factbook database (in a somewhat simplified form). For this experiment, countries of the world were divided into classes representing different fertility rate ranges. Figure 2 presents a rule characterizing 16 of the 42 countries with the smallest fertility rates (no more than 2 per woman).

Fertility ≤ 2 per woman if:

	p	*n*	*compl*	*cons*
[Birth Rate = *10..20 per 1000 people*]	42	20	100%	68%
[Religion is *R. Catholic* or *Orthodox* or *Anglican* or *Shinto*]	24	31	57%	44%
[Infant Mortality Rate ≤ *40 per 1000 babies*]	41	54	98%	43%
[Population Growth Rate ≤ *4%*]	32	56	76%	36%
[Literacy ≥ *70%]*	35	71	83%	33%
[Life Expectancy = *60..80 years*]	41	92	98%	31%
[Death Rate = *5..15 per 1000 people*]	42	102	100%	29%
[Net Migration Rate ≥ *-10 per 1000 people*]	42	140	100%	23%
Rule Total (all conditions):	*16*	*0*	*38%*	*100%*

Figure 2. Example of an annotated rule in the attributional calculus.

3 ASSOCIATION GRAPHS

The attributional rules provide details about relationships among attributes or concepts. To illustrate such relations graphically and with less detail, we developed a visualization method called *association graphs*. In an association graph, nodes represent attributes or concepts, and directed weighted links indicate relationships among nodes. The thicker the link, the stronger the relationship (based on the consistency of the attributional condition). Links are annotated by symbols indicating the type of relationship between connected nodes. A monotonically growing (decreasing) functional relationship between nodes is indicated by the symbol "+" ("–"). A functional relationship that has its maximum (minimum) in the middle of the range of the independent attribute is indicated by the symbol "^" ("v"). Links that represent relationships that do not follow to any of the above types are left unlabeled. These symbols are also used when the relationship only approximates one of the relationship classes defined above.

A rule relating several attributional conditions to another condition is represented by linking the involved conditions with an arc. For example, Figure 3 shows an association graph representing the rule from Figure 2.

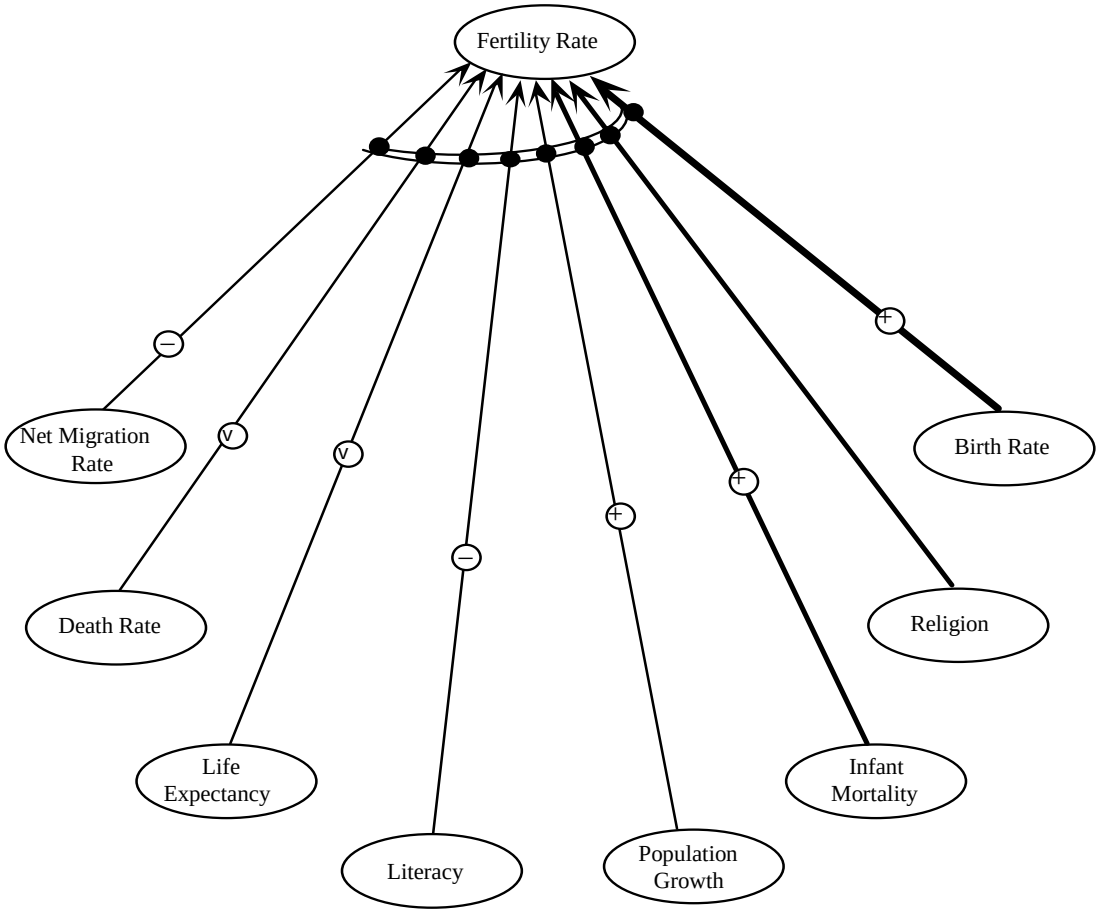

Figure 3. An association graph representing the attributional rule from Figure 2.

A somewhat similar approach to knowledge visualization is used in the CLEMENTINE system, a data mining toolkit commercially developed by Integral Systems, Ltd. A major difference between the association graphs and the representations used in CLEMENTINE is that the former are able to represent multi-argument relationships, not just binary relationships.

Another difference is that the association graphs are representations at a higher level of abstraction, because their nodes represent attributes, rather than individual attribute values, and their links can represent sets of composite conditions employed in attributional calculus, rather than only attribute-value conditions.

4 A LANGUAGE FOR DEFINING KNOWLEDGE SCOUTS: KGL-1

As mentioned above, knowledge scouts are defined by creating scripts in a knowledge generation language. Below is a brief description of our first version of such a language, called KGL-1 (Kaufman and Michalski, 1998). KGL-1 has been designed according to the following requirements:

1. The language integrates database operators, knowledge base operators, and knowledge generation operators in a single representational system.

2. Inductive inference and any other knowledge generating programs integrated in the inductive database can be invoked by individual KGL-1 operators.

3. Results from any KGL-1 operator can be used as inputs to any operator for which they are semantically applicable.

4. Parameters to be used in running any knowledge generating program can be specified as arguments of the corresponding KGL-1 operator.

5. KGL-1 statements can refer to various properties of the data in the database (For example, "If there are 10% new examples of class A in the database, invoke a rule learning operator"; "Determine the percentage of missing values in the database.")

6. KGL-1 statements can refer to the properties of generated knowledge or the background knowledge, in particular, the name, the type and the domain of attributes, the attributional rules and their components, the groups of rules (rulesets), all components of the annotations of the rules, etc. (For example, "Select nominal attributes with five or fewer values in their domain, and generate for them decision rules using all numerical attributes as independent variables," "If the completeness of the first rule in a ruleset for a given class is at least 95%, remove all remaining rules from the ruleset," or "Determine all conditions in a ruleset for a given class whose consistency is above 80% and support is between 30 and 50.")

7. Looping and branching are implemented as in conventional programming languages.

8. KGL-1 has capabilities for defining data management, knowledge management and knowledge generation tasks that may be involved in the extraction, manipulation, generation and displaying of any data or knowledge in the system. (For example, select a target dataset from a database, generate an attributional ruleset that satisfies certain criteria, and display and/or print it under given conditions).

KGL-1 has been partially implemented in the INLEN system (Michalski and Kaufman, 1998). To illustrate how KGL-1 can be used for building knowledge scouts, Figure 4 presents a simple KGL-1 script for creating a knowledge base that includes, among other things, the attributional rule shown in Figure 2, and for analyzing that knowledge. Comments are italicized and bracketed.

The log file output from the above script is shown in Figure 5. The first part of the output shows the number of strong rules, as determined by three criteria. Because the Fertility ruleset was found too complex (having more than 150 conditions), a learning process was repeated using only the four most relevant independent attributes, as determined by the operator SELECT. The last part of the output presents numbers of conditions in the ruleset for Life Expectancy that exceed different thresholds regarding the p / n ratio (support divided by disparity, assuming that the disparity is not zero; when disparity is zero, all thresholds are satisfied). The last three lines indicate one consistent condition in the ruleset for Life Expectancy.

As shown above, KGL-1 provides a unique combination of features, which is not present in other languages for automated knowledge discovery. Specifically, it supports the use of an attributional calculus representation, the employment of diverse symbolic learning and

inference methods, a tight coupling with the annotated rulesets, and the ability to build advanced knowledge scouts. Most existing languages for data exploration use a Prolog-based approach. One exception is M-SQL, which extends the SQL data query language by adding to it the ability to query for certain types of rules and to invoke an association rule generating operator (Imielinski, Virmani, and Abdulghani, 1996). KGL-1 differs from M-SQL in that it is able to define complex data mining plans that involve many different types of knowledge generation operators, and more closely resembles a programming language than a query language.

```
open PEOPLE                                    {Select PEOPLE database}
do CHAR(decision=all, pfile=people1.lrn)       {Characterize concepts
                                               representing single values
                                               of all attributes, using
                                               parameters specified in file
                                               people1.lrn}
strongPGrules1 = #rules(PGR, compl >= 60)      {Count rules for Population}
strongPGrules2 = #rules(PGR, supp >= 25)       {Growth Rate that satisfy}
strongPGrules3 = #rules(PGR,                   {three different conditions}
   num_conds(cons >= 50 and supp > 10) > 2){for threshold of strength}
print "Number of strong PGR rules:
   Type 1 = ", strongPGrules1, ",
   Type 2 = ", strongPGrules2, ",
   Type 3 = ", strongPGrules3
if #conditions(Fert) > 150                     {Is Fert ruleset too
                                               complex?}
  begin
  do SELECT(attributes, decision=Fert,
     thresh=4, out=PEOPLE2, criterion=max) {If so, find "thresh" best}
  do CHAR(pfile=people1.lrn, decision=Fert){independent attributes,
then
  end                                          {recharacterize}
for i = 1 to 6
begin                                          {For each value of i, 1-6,}
print "Number of LE conditions with p/n       {count & display number of}
       ratio of at least", i, ":1 =",          {Life Expectancy conditions}
      #conditions(LE, cons >= i/(i+1))         {with consistency • i/(i+1)}
end
```

Figure 4. A KGL-1 script for defining a knowledge scout exploring a demographic database.

```
Number of Strong PGR rules: Type 1 = 1, Type 2 = 1, Type 3 = 7
Selecting best attributes from PEOPLE for concept Fert –
      Attributes chosen:
      Birth Rate, Predominant Religion, Life Expectancy, Death Rate
Number of LE Conditions with p/n ratio of at least 1:1 = 25
Number of LE Conditions with p/n ratio of at least 2:1 = 10
Number of LE Conditions with p/n ratio of at least 3:1 = 5
Number of LE Conditions with p/n ratio of at least 4:1 = 1
Number of LE Conditions with p/n ratio of at least 5:1 = 1
Number of LE Conditions with p/n ratio of at least 6:1 = 1
```

Figure 5. Output from the KGL fragment from Figure 4.

A language that takes a different approach to the acquisition of knowledge is KQML (Finin et al. 1994). KQML is viewed as a tool by which intelligent agents may communicate among themselves and exchange the information needed to complete their individual tasks. It is thus designed to permit queries for individual pieces of knowledge. KGL differs from KQML in

that it focuses more on queries for knowledge that fits a given abstract template (e.g., rules of a certain degree of strength), and that within the language one can call out to diverse learning and discovery operators to generate the knowledge base to be used.

The CLEMENTINE system allows a user to specify a plan for a sequence of actions by a simple interface. KGL differs in that the language allows the specification of branching and looping conditions that may be based on the knowledge base or on locally assigned variables, and in employing different, more diverse, and in some cases significantly more powerful knowledge generation operators.

5 STUDY 1: A KNOWLEDGE SCOUT FOR DETERMINING RELATIONSHIPS IN A MEDICAL DATABASE

Among major means for improving medical decision making and preventing diseases is to develop more advanced models of the relationships among medical conditions, manifestations, lifestyles, and therapies. Such models must be able to represent multidimensionality of relations, for example, that a confluence of several factors may be needed to develop a given disease. Such relations are difficult to capture by statistical correlations or covariances among individual factors.

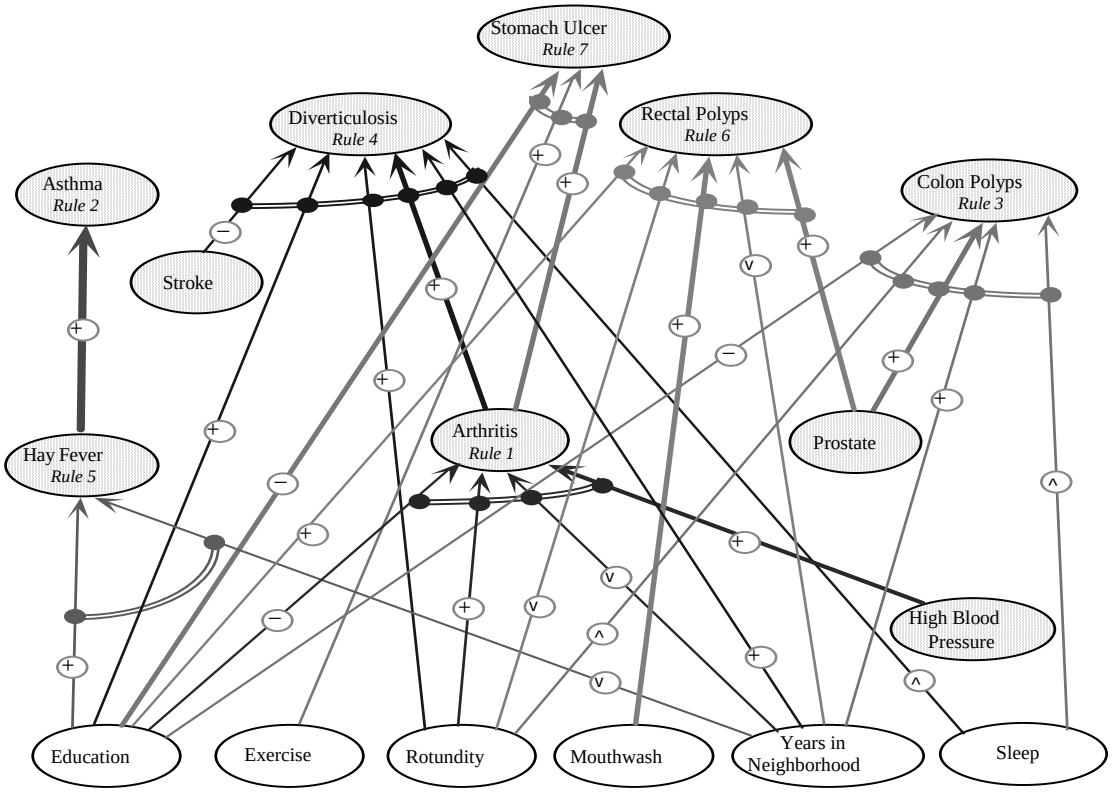

Figure 6. An association graph linking a group of diseases with patient characteristics determined from a subset of the ACS Second Cancer Prevention Study database.

Our first efforts toward building such models involved developing a knowledge scout that searches for strong patterns in a database representing facts about diseases, manifestations and lifestyles, which the American Cancer Society's Second Cancer Prevention Study (CPS-II). For our preliminary experiments, we selected a small subset of records (73,553 records from 1.2 million), the subset pertaining to male non-smokers, ages 50-65. Each patient was characterized in terms of such attributes as "rotundity" (a function of the patient's height and weight), the amount of exercise, the number of hours of sleep, the education level, the use of mouthwash, etc. Each record had background information on the respondent, as well as information on whether or not he had occurrences of any of the 25 types of disease. The pilot study involved an application of an inductive rule learning operator to generate multidimensional patterns that link factors with individual diseases. Many patterns were generated. Figure 6 presents a collection of interrelated patterns, using a *concept association graph* (Section 3).

This study was preliminary, and should not be taken as new medical knowledge. However, these early results show a significant potential of the proposed methodology. Generated models can provide new insights into relationships between diseases and lifestyles, and assist doctors in the disease diagnosis and treatment. They can also serve as guides to patients for disease prevention.

6 STUDY 2: A KNOWLEDGE SCOUT FOR DETERMINING PATTERNS IN PARENT-CHILD DATABASE

This study used the National Youth Survey database (Elliott, 1976). Students from across the United States were interviewed along with their parents or other adult guardians. The dataset contained 1735 records, about 30% of whose approximately 600 attributes represented responses by the adult, 65% of which represented responses by the student, and the rest represented environmental factors. A knowledge scout was created by writing a KGL-1 script for generating rules for the 415 discrete-valued (either nominal or linearly ordered) attributes taken as decision attributes, after having reduced the dataset by invoking an operator to select the 25 attributes most likely relevant to the decision attribute, and then projecting the data accordingly. The KGL program (Figure 7) then selected the strongest rules (those with a completeness over 80%, or a completeness over 60%, while covering over 50 training examples.

```
begin
for i = 0 to 414
  begin
  open SURVEY
  do SELVAR(decision=i, out=SURVEY2, thresh=25, criterion=avg)
  do DISCSET(decision=0, scope=1, compile=no)
  end
end
open SURVEY2
for i = 0 to 414
  begin
  forall rules(i, compl > 80 or (compl > 60 and supp > 50))
     print "decision =", i, "  class = ", class, "  rno = ", rno
  end
end
```

Figure 7. A KGL-1 program for survey data exploration.

Below is an example of one of many rules generated and cited as a strong rule. It describes the class of responses in which the responding adult said the student did not get into trouble with the law. There were 1618 responses in this class, and 65 in the negative class (adults who said the student did get into trouble with the law).

Adult says student does not get into trouble with the law if:

	p	*n*	*compl*	*cons*
[Student's frequency of lying about age ≤ *1 per week*]	1512	52	93%	96%
[Student says stealing over $50 is *wrong* or *very wrong*]	1587	60	99%	96%
[Student hired prostitute in past year ≤ *twice*]		1603	63	99% 96%
[Student stole car in past year ≤ *twice*]	1612	64	99%	96%
Rule Total:	*1376*	*54*	*85%*	*96%*

The above rule, which is self-explanatory, can be viewed as an answer to a query to create a characterization of students that their guardians believe do not get into trouble with the law. This study shows that the proposed line of research can provide a useful extension of database technology.

7 SUMMARY AND FUTURE RESEARCH

This paper presented a novel approach in combining advanced learning, inference and representation techniques in order to build knowledge scouts, personal intelligent agents that can assist in complex data mining tasks by focusing on the types of patterns that are likely to be of interest to a given user. These scouts can be applied as part of an inductive database, a system that integrates data and knowledge base technologies with inductive inference and discovery.

The use of association graphs provides a visualization of general patterns in data and knowledge that is easy to understand. Similarly, expressing rules in the attributional calculus provides for understandable knowledge that can be annotated with various weights that describe characteristics of the knowledge. A KGL-1 script is able to call upon multiple inference and data exploration operators to create, access and query such knowledge.

The initial version of the KGL-1 language is implemented within the INLEN system, and is tightly coupled to its data and knowledge structures. A planned topic for further development is to create a general version of the language that can be applied to other knowledge structures, while broadening its expressiveness.

We have alluded to the task of user modeling and updating a profile over time. Future research will explore the topic of adapting techniques of learning and maintaining patterns that change over time and applying them to this problem.

REFERENCES

Agrawal, R., Imielinski, T. and Swami, A., Mining Association Rules between Sets of Items in Large databases, *Proceedings of the ACM SIG-MOD Conference on Management of Data*, 207-216, Washington, D.C., 1993.

Elliott, D., "National Youth Survey (United States), Wave I, 1976," [Computer file]. ICPSR version, University of Colorado Behavioral Research Institute, Boulder, CO, 1977, distributed by Inter-University Consortium for Political and Social Research, Ann Arbor, MI, 1994.

Finin, T., Fritzson, R., McKay, D. and McEntire, R., "KQML as an Agent Communication Language," *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*., ACM Press, 1994.

Imielinski, T., Virmani, A. and Abdulghani, A., "DataMine: Application Programming Interface and Query Language for Database Mining," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 256-261, 1996.

Kaufman, K. and Michalski, R.S., "Discovery Planning: Multistrategy Learning in Data Mining," *Proceedings of the Fourth International Workshop on Multistrategy Learning*, Desenzano del Garda, Italy, pp. 14-20, 1998.

Kaufman, K. and Michalski, R.S., "Learning From Inconsistent and Noisy Data: The AQ18 Approach," *Proceedings of the Eleventh International Symposium on Methodologies of Intelligent Systems*, Warsaw, 1999.

Michalski, R. S., "Synthesis of Optimal and Quasi-Optimal Variable-Valued Logic Formulas," *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, pp. 76-87, 1975.

Michalski R. S. and Kaufman, K., "Data Mining and Knowledge Discovery: A Review of Issues and Multistrategy Methodology," in Michalski, R.S., Bratko, I. and Kubat, M. (eds.), *Machine Learning and Data Mining: Methods and Applications*, London, John Wiley & Sons pp. 71-112, 1998.

Michalski, R.S., "NATURAL INDUCTION: A Theory and Methodology of the STAR Approach to Symbolic Learning and Its Application to Data Mining," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 1999.