Proceedings of the Fourth Int'l Conference on Flexible Query Answering Systems, FQAS 2000, Warsaw, Poland, pp. 485-496, October, 2000.

A Knowledge Scout for Discovering Medical Patterns: Methodology and System SCAMP

Kenneth A. Kaufman[1] and Ryszard S. Michalski1[1,2]

[1] Machine Learning and Inference Laboratory, George Mason University, Fairfax VA 22030, USA
[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. Knowledge scouts are software agents that autonomously synthesize knowledge of interest to a given user (target knowledge) by applying inductive database operators to a local or distributed dataset. This paper describes briefly a method and a scripting language for developing knowledge scouts, and then reports on experiments with a knowledge scout, SCAMP, for discovering patterns characterizing relationships among lifestyles, symptoms and diseases in a large medical database. Discovered patterns are presented in two forms: (1) attributional rules, which are expressions in attributional calculus, and (2) association graphs, which graphically and abstractly represent relations expressed by the rules. Preliminary results indicate a high potential utility of the presented methodology for deriving useful and understandable knowledge.

1 Introduction

When applying data mining tools to a large database, a user may have to perform many repetitions and trials of various operations before desired knowledge is discovered. This process can be difficult and time-consuming. Such a situation occurs, for example, in the PC-based multistrategy data mining and knowledge discovery system, INLEN [8][16][18] Researchers have been addressing this problem by adding new operators to existing query languages (e.g., [7][12][22]), or by building a meta-language that integrates a query language with various knowledge discovery operators (e.g., [9][19]).

Another challenge in data mining is how to specify *target knowledge*, that is, knowledge that is likely to be of interest to a given user. Obviously, such knowledge cannot be defined precisely, as the whole purpose of the search is to find something new and unexpected. Furthermore, the target knowledge may be changing over time, as it depends on the current goals and knowledge of the user. This indicates a need for a mechanism to acquire and monitor a profile of the user's interests, and apply this profile in the search for target knowledge.

To address the problems outlined above, the idea of a *knowledge scout* has been proposed. A knowledge scout is a software agent that employs resources of an *inductive database*, in order to search for and synthesize target knowledge. The concept of an inductive database does not have one commonly agreed meaning (e.g., [3][11][14][19]). Here, by an inductive database we mean a system that

integrates a database with inductive inference capabilities [14], allowing answers to queries asking for *plausible knowledge*, (knowledge not directly or deductively obtainable from the database, but hypothesizable through inductive inference [19]). This knowledge can be in the form of hypotheses about future datapoints, expected consequences, generalized data summaries, emerging global patterns, exceptions from hypothesized patterns, suspected errors and implied inconsistencies, hypothetical plans synthesized from the data, etc. [6][7][15][17].

A general diagram of an inductive database is presented in Figure 1.

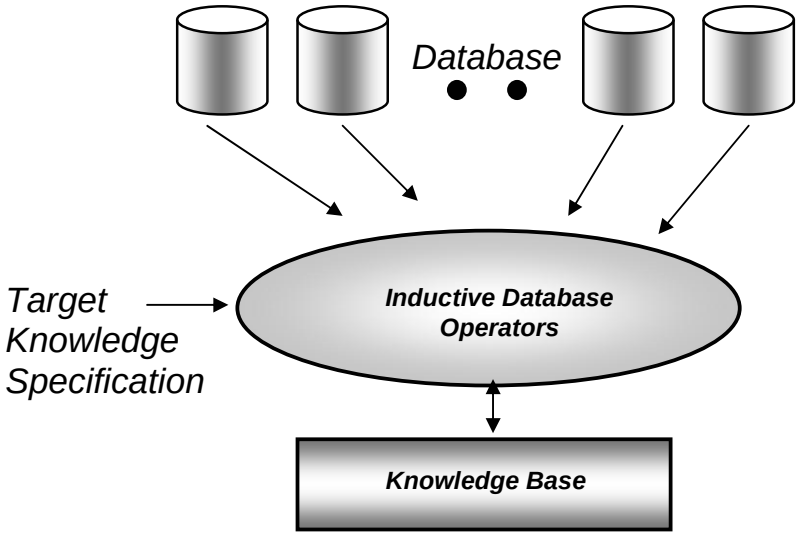

Fig. 1. A general diagram of an inductive database.

An inductive database implements database operators based on inference methods developed in the fields of machine learning, statistics, and uncertain reasoning. These operators are integrated with conventional database operators through a *knowledge generation language (KGL)*. In addition, an inductive database includes a *knowledge base* that contains meta-knowledge, domain constraints, models of users' interests, etc. Using KGL, one can implement multiple knowledge scouts, dedicated to pursuing different target knowledge. A KGL script that defines a knowledge scout includes a plan of operations to be performed on the database and an abstract definition of the target knowledge.

The target knowledge for a knowledge scout is defined abstractly by specifying properties of pieces of knowledge that are likely to be of interest to the given user (or a group of users). For example, a target knowledge may be defined as "patterns that relate variables from the set T to those in the set S", or "patterns that achieve the highest score on a given pattern quality measure" (e.g., [10]), or "a data classification scheme that maximizes a criterion of clustering quality" (e.g., [20]).

In order to synthesize target knowledge, a knowledge scout may execute long sequences of operations involving data, intermediate results, and background

knowledge. At every step, an operator's application may depend on previous results. The user's interests and relevant knowledge are partially defined a priori, and partially updated during the scout's lifetime. As inductively derived knowledge generally has lower certainty than directly or deductively obtained knowledge, results of inductive queries are annotated by certainty measures.

In this paper, we focus on a medical application of knowledge scout technology. The following sections present briefly a methodology for building knowledge scouts, and then concentrate on the application of the methodology to the development of a knowledge scout, called SCAMP (Scout for Acquiring Medical Patterns). SCAMP searches for multidimensional patterns characterizing medical conditions, manifestations, lifestyles, and therapies. Such patterns may capture multi-argument relations, in which a confluence of several medical factors indicates a given disease, while the presence of any single one may not.

For testing these ideas, we used a database representing facts about diseases and lifestyles, developed by the American Cancer Society's Second Cancer Prevention Study (CPS-II). In the experiments described here, we used a set of 73,553 records that pertain to male non-smokers, age 50-65. Patients are characterized by attributes such as "rotundity" (a function of height and weight), the amount of exercise, the number of hours of sleep, the education level, the use of mouthwash, etc. Together with these characteristics, there is information indicating whether or not the patient had any of 25 types of disease.

The following sections describe two knowledge representation systems used for representing patterns and multidimensional relationships by a knowledge scout. The first, *attributional calculus*, is a rule-based representation language for a rich, but simple, expression of knowledge. The second, *association graphs*, allows for the easy visualization of relationships among concepts, including multi-argument ones. We then present KGL-1, a prototype language for defining knowledge scouts, and describe SCAMP, along with the preliminary results it has achieved.

2 Attributional Rules

The language in which patterns 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, making their discovery difficult. If the language is too rich, the pattern search space may become computationally prohibitive. In addition, for many applications it is important that patterns are easy to understand and interpret (the comprehensibility postulate [15]). Guided by such considerations, we employ *attributional calculus rules* for expressing patterns or knowledge of interest [17].

The attributional calculus is an extension of propositional calculus in which literals are replaced by *attributional conditions*. Such conditions represent relational statements that bind attributes to a set of their values or other attributes (see below). 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 [13].

An attributional condition is expressed in the form: **[L rel R]**, where **L** is an attribute, or one or more attributes with the same domain joined by "&" or "v"; **R** is a value or a list of values joined by *internal disjunction*, a pair of values joined

by *range*, or an attribute with the same domain as the attribute(s) in **L**; and **rel** is a relational symbol. For illustration, the following are examples and explanations of attributional conditions. Note that these conditions are simple to interpret and translate into equivalent natural language expressions.

[blood-pressure = normal] *(the blood pressure is normal)*
[income = 20K..30K] *(the income is between 20K and 30K)*
[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 an 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) [17]. The association rules presented in [1] could be viewed as a specialized form of PARs. Attributional rules that characterize a pattern in a database can be determined using an inductive operator based on the AQ-18 rule learning program [9]. Such an operator generates rules with annotations specifying the *support*, *disparity*, *completeness* and *consistency* for each condition in the rule, and for each rule as a whole.

The support of a condition (or, a rule), denoted by *p*, is defined as the number of tuples representing a given relationship ("positive examples") that satisfy the condition (rule). The disparity, denoted by *n*, is defined as the number of "negative examples" that satisfy the condition (rule). Given that the training dataset has *P* positive and *N* negative examples, we make these definitions:

The **completeness** of a condition or rule, denoted *compl*, is equal to p / *P*.

The **consistency** of a condition or rule, denoted *cons*, is equal to *p* / (*p* + *n*).

The **consistency of randomly guessing**, denoted *cguess*, is *P* / (*P* + *N*).

The program also generates other annotations, such as exceptions, ambiguity, and description quality (e.g., [10]), which are beyond the scope of this paper.

To illustrate, Figure 2 presents an attributional rule generated by SCAMP from a dataset consisting of 7351 examples, of which *P*=2063 represented individuals who suffered from high blood pressure, and *N*=5288 represented those who didn't. The rule states that patients with high blood pressure are characterized by having high or very high rotundity, an educational level that includes high school and possibly some college, and exercise at a medium level or less.

High_Blood_Pressure is present if:	p	n	compl	cons
[Rotundity ≥ high]	689	1058	33%	39%
[Education_Level is hs_grad..some_college]	1055	2213	51%	32%
[Exercise ≤ medium]	1838	4473	89%	29%
Rule Total (all conditions):	303	332	15%	48%

Fig. 2. Example of an attributional rule with annotations

Note that in this rule, each condition separately has a relatively low consistency (between 29% and 39%). When all three conditions are combined, the consistency jumps to 48%, which is significantly higher than randomly guessing the positive class (*cguess* = 28%).

3 Association Graphs

Attributional rules characterize relationships among attributes (or concepts) through a logic-style expression, which can be easily translated to natural language. To provide a user with a simpler, more abstract way of representing such relationships, we have developed a visualization method called *association graphs*, whose nodes represent attributes or concepts, and inter-node links characterize relationships among nodes. The links are directed, weighted and annotated. The direction indicates the direction of the relationship. The weight (represented by the thickness of the link) indicates the strength of 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 variables is indicated by the symbol "+" ("–") attached to the link between corresponding nodes. A functional relationship that has its maximum (minimum) in the middle of the range of the independent attribute is indicated by the symbol "^" ("v").

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

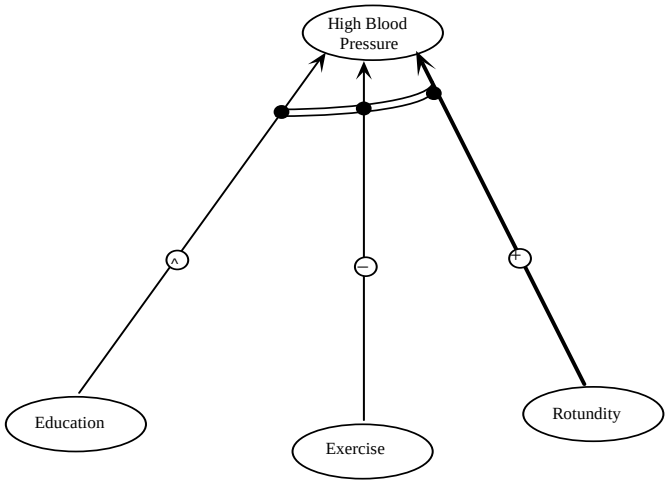

Fig. 3. An association graph representing the attributional rule from Figure 2.

Association graphs can represent complex multivariate relationships in a simple fashion. They provide an advanced tool for knowledge visualization that differs from some used in data mining systems (e.g., in CLEMENTINE, a data mining toolkit commercially developed by Integral Systems, Ltd.). One major difference is that the presented association graphs can represent multi-argument relations, not only binary relations. Another difference is that they are representations at a higher abstraction level. Specifically, their nodes represent attributes, rather than

individual attribute values, and links represent composite conditions employed in attributional calculus, rather than only attribute-value conditions.

4 A Metalanguage for Defining Knowledge Scouts: KGL-1

Knowledge scouts are defined by creating scripts in the knowledge generation language. Below is a brief description of our first version of such a language, KGL-1 [9]. 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 programs, as well as other knowledge processing programs integrated in the inductive database can be invoked by single 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 to the corresponding KGL-1 operator.
5. KGL-1 statements can refer to various properties of the data in the database.
6. KGL-1 statements can refer to various properties of generated knowledge or the background knowledge, in particular, to attribute values, to the type and the domain of any attribute, the attributional rules and their components, to the groups of rules (rulesets), to any component of the annotations of the rules, etc.
7. Looping and branching are implemented, as in many programming languages.
8. The language can invoke data management, knowledge management and knowledge generation operators that may be involved in the extraction, manipulation, generation and display of any data or knowledge in the system

The KGL-1 metalanguage presented above provides a unique combination of features not present in other languages for automated knowledge discovery. Many current languages use a Prolog-based approach, and have quite limited types of knowledge generation operators available. Most exceptions to the Prolog-based approach are SQL-oriented, extending the data query language by adding an ability to query for certain types of rules and invoke association rule generation (e.g., [7][12]). KGL-1 differs from these in that one may define complex data mining plans that involve many types of knowledge generation operators; it more closely resembles a programming language than a query language.

A language somewhat related to KGL-1 is KQML, which provides means by which agents may communicate among themselves and exchange task-relevant information [4]. Another related language is used in CLEMENTINE, which allows a user to specify a plan for a sequence of actions by a simple interface.

A prototype version of KGL-1 has been implemented in the INLEN-3 system [9]. Each operator manipulates the database and/or knowledge base, and contains arguments that identify its input, output, and the parameters that deviate from the defaults. These parameters make it possible to specialize an operator to multiple forms; each operator thus corresponds to a set of data/knowledge transformations.

The following operators have been integrated into KGL-1, or are in the process of integration through the adaptation of already implemented programs:

- **CHAR**(Datatable, Class, Params): Characterize the entities in the Datatable that belong to the Class, by inducing their characteristic description [15].
- **DIFF**(Datatable, Class1, Class2, Params): Differentiate entities in Class1 from Class2 in the Datatable, by inducing a discriminant description [15].
- **SELECT**(Target, Datatable, Params): Select components from the Datatable, whose type is defined by Target according to the method specified in Params. The Target determines whether attributes or examples are to be selected.
- **TEST**(Datatable, Ruleset, Params): Test the Ruleset's knowledge against a set of testing examples in the Datatable. Each testing example is classified based on the rule it best matches, using a strict or a flexible matching method [2][21]. The operator generates a report on the ruleset's predictive accuracy.
- **CLASSIFY**(Examples, Ruleset, Params): Assign the Examples to classes using the Ruleset. This operator invokes an inference procedure that applies rules in the Ruleset to Examples. The output of this operator includes a list of Examples, their classification and a measure of certainty that that classification.
- **CLUSTER**(Datatable, Params): Split records in the Datatable into a set of conceptual clusters. The operator is based on the conceptual clustering program CLUSTER2 [5][20], It defines clusters by attaching a column to the Datatable whose indices indicate clusters, and describes each one by an attributional rule.
- **GENSTAT**(Datatable, Params): Determine and report statistical characteristics of the Datatable, such as means, modes and variances for attributes in subsets of data associated with different target variables. It can also generate covariances and correlation coefficients between numerical attributes.
- **VISUALIZE**(Input, Params): Visualize the items specified in the Input, using an association graph or diagrammatic visualization method [23].

To illustrate how KGL-1 is used for building knowledge scouts, Figure 4 presents a script for a simple knowledge scout that creates and examines a knowledge base of relationships among attributes in a medical database. Each such relationship is expressed by a set of attributional rules, generated by the CHAR operator. One of these rules was illustrated in Figure 2.

The log file output from the above script is shown in Figure 5. The first part of the output shows the number of strong attributional rules, as determined by three different criteria imposed on the rule strength. No rules satisfied the first (50% completeness) or third (two conditions with both 50% consistency and 300 example support) criteria. The second criterion (rules with support greater than 25) was met by two of the six generated rules. Because the High Blood Pressure ruleset was not found to be too complex (having fewer than 50 conditions), the specified simplification-through-relearning process was not applied. The last part of the output presents numbers of conditions in the ruleset for Asthma that exceed different thresholds regarding the *p / n* ratio (that is, support divided by disparity). The last output indicates that there was only one condition with p/n ratio greater or equal to 1:3 in the Asthma ruleset, and one other with a ratio of at least 1:5.

Summarizing, KGL-1 supports a powerful knowledge representation, employs a wide range of learning and inference operators, operates on components of the knowledge base and the database, and provides mechanisms for implementing advanced knowledge scouts. The attributional rules allow the system to compactly and understandably represent complex multidimensional relationships.

```
open ACSDATA                                    {Select ASCDATA database}
do CHAR(decision=all, pfile=ACS1.lrn)           {Characterize concepts
                                                 representing values
                                                 of all attributes using
                                                 parameters specified in
                                                 file ACS1.lrn}
strongArtRules1 = #rules(Arth, compl >= 50){Count rules for}
strongArtRules2 = #rules(Arth, supp >= 25) {Arthritis that satisfy}
strongArtRules3 = #rules(Arth,             {3 different conditions}
 num_conds(cons >= 50% and supp > 300) > 1){for threshold of
                                            strength}
print "Number of strong Arthritis rules:
 Type 1 = ", strongArthRules1, ",
 Type 2 = ", strongArthRules2, ",
 Type 3 = ", strongArthRules3
if #conditions(HBP) > 50                        {Is High Blood Pressure}
  begin                                         {ruleset too complex?}
  do SELECT(attributes, decision=HBP,
     thresh=15, out=ACS2, criterion=max)        {If so, find "thresh"}
  do CHAR(pfile=ACSimplify.lrn,                 {best independent}
     decision=HBP)                              {attributes, then}
  end                                           {recharacterize}
else
  print "HBP Ruleset sufficiently simple"
for i = 1 to 6
begin                                           {For each value of i from}
print "Number of Asthma conditions with        {1 to 6, count and show}
     p/n ratio of at least 1: ", i, " =",       {number of Asthma}
     #conditions(Asth, cons >= 1/(i+1))         {conditions with}
end                                             {consistency ≥ 1/(i+1).}
```

Fig. 4. A KGL-1 script for defining a knowledge scout exploring the medical database.

```
Number of Strong Asthma rules: Type 1 = 0, Type 2 = 2, Type 3 = 0
HBP Ruleset sufficiently simple
Number of Asthma Conditions with p/n ratio of at least 1:1 = 0
Number of Asthma Conditions with p/n ratio of at least 1:2 = 0
Number of Asthma Conditions with p/n ratio of at least 1:3 = 1
Number of Asthma Conditions with p/n ratio of at least 1:4 = 1
Number of Asthma Conditions with p/n ratio of at least 1:5 = 2
Number of Asthma Conditions with p/n ratio of at least 1:6 = 2
```

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

5 SCAMP--A Knowledge Scout for Discovering Medical Patterns

This section describes briefly SCAMP, a medical knowledge scout for exploring the ACS dataset. Our experiments were done with SCAMP specified by the following script:

For each disease attribute in the dataset:

Select randomly, about 10% of the data for training.

Determine a ruleset consisting of strong patterns discriminating cases in which the disease is present from other cases

From the generated rulesets, maintain only rules whose support levels are within 40% of the strongest one in the ruleset.

In total, during the course of these experiments, over 10,000 patterns were generated, some strong, and many spurious. To illustrate the interaction among these patterns, seven of the stronger rules were combined into an association graph (Figure 6). Shaded nodes represent diseases, and unshaded ones represent other factors provided in the data. Differences in line thicknesses, indicating the relative informational significance of individual conditions, are evident.

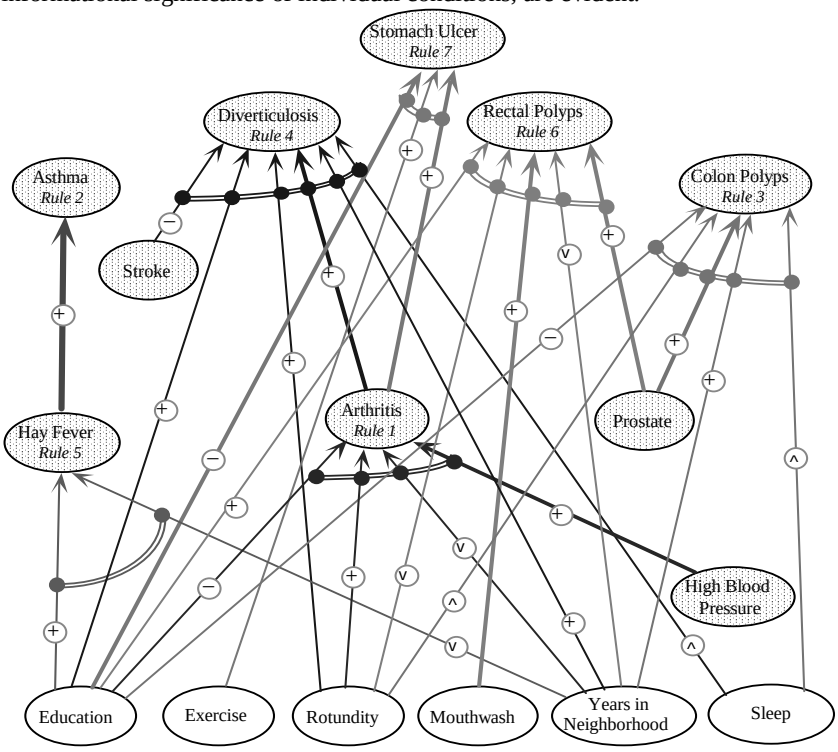

Fig. 6. An association graph linking a group of diseases with patient characteristics, as determined by SCAMP from a subset of the ACS Second Cancer Prevention Study database.

This association graph is an illustrative presentation of the relationships among attributes characterizing patients' lifestyles and diseases. One can easily see the lifestyle characteristics associated with different diseases, and the type of influence of each characteristic on various diseases. This is, however, only a preliminary result that does not intend to serve as a contribution to medical science. Nevertheless, it indicates a strong potential of the presented methodology for discovering and modeling useful and novel medical patterns.

6 Summary and Future Research

This paper described briefly a methodology for integrating machine learning and inference methods with database operators for the purpose of automatically conducting complex data mining and knowledge discovery operations. The central idea in this methodology is a knowledge scout, defined as a software agent that utilizes resources of an inductive database to search for and synthesize target knowledge. A knowledge scout is defined by a script in a knowledge generation language. An initial version of such a language, KGL-1, implemented in the INLEN-3 inductive database system, has been briefly described.

A knowledge scout, SCAMP, has been developed for conducting large-scale experiments in a medical database. Two knowledge representations, attributional rules and association graphs, were described and illustrated by selected discovered patterns from this dataset. Association graphs can provide 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. The preliminary results achieved by SCAMP indicate a high potential utility of this methodology.

Acknowledgments

The authors thank Jim Logan for providing the American Cancer Society database and discussing experiments done in Study 1. This research was conducted in the Machine Learning and Inference Laboratory at George Mason University under partial support from the National Science Foundation under Grants No. IIS-0012121, IIS-9904078 and IRI-9510644.

References

1. Agrawal, R., Imielinski, T. and Swami, A. (1993) Mining Association Rules between Sets of Items in Large Databases. Proceedings of the ACM SIGMOD Conference on Management of Data, 207-216.
2. Bergadano, F., Matwin S., Michalski, R.S. and Zhang, J. (1992) Learning Two-tiered Descriptions of Flexible Concepts: The POSEJDON System. Machine Learning 8, 5-43.
3. Boulicaut, J., Klemettinen, M. and Mannila, H. (1998) Querying Inductive Databases: A Case Study on the MINE RULE Operator. Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98).
4. Finin, T., Fritzson, R., McKay, D. and McEntire, R. (1994) KQML as an Agent Communication Language. Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), ACM Press.
5. Fischthal, S. (1997) A Description and User's Guide for CLUSTER/2C++ A Program for Conjunctive Conceptual Clustering. Reports of the Machine Learning and Inference Laboratory, MLI 97-10, George Mason University, Fairfax, VA.
6. Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B. and Zaiane, O.R. (1996) DBMiner: A System for Mining

Knowledge in Large Relational Databases. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 250-255.
7. Imielinski, T., Virmani, A. and Abdulghani, A. (1996) DataMine: Application Programming Interface and Query Language for Database Mining. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 256-261.
8. Kaufman, K. (1997) INLEN: A Methodology and Integrated System for Knowledge Discovery in Databases. Ph.D. dissertation, George Mason University, Fairfax, VA.
9. Kaufman, K. and Michalski, R.S. (1998) Discovery Planning: Multistrategy Learning in Data Mining. Proceedings of the Fourth International Workshop on Multistrategy Learning, 14-20.
10. Kaufman, K. and Michalski, R.S. (1999) Learning From Inconsistent and Noisy Data: The AQ18 Approach. Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems.
11. Mannila, H. (1997) Inductive Databases and Condensed Representations for Data Mining. in Maluszynski, J. (ed.), Proceedings of the International Logic Programming Symposium, MIT Press, Cambridge.
12. Meo, R., Psaila, G. and Ceri, S. (1996) A New SQL-like Operator for Mining Association Rules. Proceedings of the 22nd VLDB Conference.
13. Michalski, R. S. (1975) Synthesis of Optimal and Quasi-Optimal Variable-Valued Logic Formulas. Proceedings of the 1975 International Symposium on Multiple-Valued Logic, 76-87.
14. Michalski, R. S. (1976) Class notes for the course on Databases, Computer Science Department, University of Illinois at Champaign-Urbana.
15. Michalski, R. S. (1983) A Theory and Methodology of Inductive Learning. In Michalski, R.S. Carbonell, J.G. and Mitchell, T.M. (eds.), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing, Palo Alto, 83-129.
16. Michalski, R.S. (1997) Seeking Knowledge in the Deluge of Facts. Fundamenta Informaticae 30, 283-297.
17. Michalski, R.S. (2000) NATURAL INDUCTION: Theory, Methodology and Applications to Machine Learning and Knowledge Mining. Reports of the Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA (to appear).
18. Michalski R. S. and Kaufman, K. (1998) 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, John Wiley & Sons, London, 71-112.
19. Michalski, R.S. and Kaufman, K. (2000) Building Knowledge Scouts Using KGL Metalanguage. Fundamenta Informaticae 40, 433-447.
20. Michalski, R.S. and Stepp, R. (1983) Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy. IEEE Trans. on Pattern Analysis and Machine Intelligence 5, 396-410.
21. Reinke, R.E. (1984) Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System. Master's Thesis, Department of Computer Science, University of Illinois, Urbana, IL.
22. Sarawagi, S., Thomas, S. and Agrawal, R. (1998) Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. Proceedings of the ACM SIGMOD International Conference on Management of Data.
23. Zhang, Q. and Michalski, R.S. (2000) KV: A Knowledge Visualization System Employing General Logic Diagrams. Reports of the Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA (to appear).