Applying a Data Miner to Heterogeneous Schema Integration

Son Dao **Brad Perry**
Information Sciences Laboratory
Hughes Research Laboratories
Malibu, CA 90265
{son,perry}@isl.hrl.hac.com

Abstract

An application of data mining techniques to heterogeneous database schema integration is introduced. We use attribute-oriented induction to mine for characteristic and classification rules about individual attributes from heterogeneous databases. Each mining request is conditioned on a subset of attributes identified as "common" between the multiple databases. We develop a method to compare the rules for two or more attributes (from different databases) and use the similarity between the rules as a basis to suggest similarity between attributes. *As a result, we use relationships between and among entire sets of attributes from multiple databases to drive the schema integration process.* Our initial efforts and prototypes applying data mining to assist schema integration prove promising and, we feel, identify a fruitful application area for data mining research.

Keywords: schema integration, multi-database interrelationships, attribute similarity, data mining, attribute-oriented induction.

Introduction

A large organization may have hundreds or even thousands independently developed and autonomous databases. US West reports having 5 terabytes of data managed by 1000 systems, with customer information alone spread across 200 different databases (Drew *et al.* 1993). In such a multidatabase environment, the sharing and exchange of information among the semantically heterogeneous components is often desired. A federated architecture for database systems has been recognized as one of several settings in which we can consider the semantic heterogeneity problem (Drew *et al.* 1993). It is also pointed out in that a key aspect of identifying and semi-automatically resolving semantic heterogeneity involves making semantics explicit.

One important problem in identifying and resolving semantic heterogeneity is to determine equivalent attributes between component databases. Research in this field has usually been carried out in pairwise comparing fashion. However, *incremental and multi-attribute relationships across database boundaries* have not been considered, and the semantics behind data is hardly revealed.

There is a new and active research area in database community whose aim is to discover knowledge hidden in huge amounts of data: *Data Mining*. Useful data patterns are to be discovered that are beyond the structural level. We believe that integration semantics can be naturally expressed via relationships among multi-database attributes. The incorporation of data mining techniques to identify these relationships is a natural approach and will provide more powerful solutions to the semantic integration problem.

We describe our initial efforts for exploiting data mining techniques in the difficult problem of semi-automated heterogeneous schema integration. In this paper we analyze how knowledge discovered through attribute-oriented induction (Cai, Cercone, & Han 1991) can be used to assist schema integration.

Background

A framework for schema integration involves identifying a representation basis, inter-database relationships, and schema conforming/restructuring rules (Batini, Lenzerini, & Navathe 1986). If we can identify all attributes from heterogeneous databases as being {equivalent, superclass, subclass, sibling, or incompatible} then complete, automated integration can occur. Thus, one method of integration relies on identifying semantic relationships between attributes. One approach is to compare attribute names (Hayne & Ram 1990) and determine degrees of similarity from a lexicon of synonyms. Another approach is to compare structural information, or meta-data, (Navathe & Buneman 1986; Larson, Navathe, & Elmasri 1989); although no solid theoretical foundation yet exists to compute degrees of similarity from meta-data. Comparing data at the content level is also employed by using statistics such as mean, variance, and coefficient of variance for individual attributes (Li & Clifton 1994). In all approaches, attributes are compared in a pairwise fashion, at best, to decide their equivalence. *Relationships between and among entire classes of attributes from multiple databases have not been utilized.*

Data mining is defined as "the nontrivial extraction of implicit, previous unknown, and potentially useful information from data" (Piatetsky-Shapiro & Frawley 1991). Various techniques for data mining have been suggested: each using a specialized set of input requirements and producing a specialized form of knowledge. To apply data mining to schema integration requires identifying a specific class of mining techniques and then exploiting the knowledge generated.

General Architecture

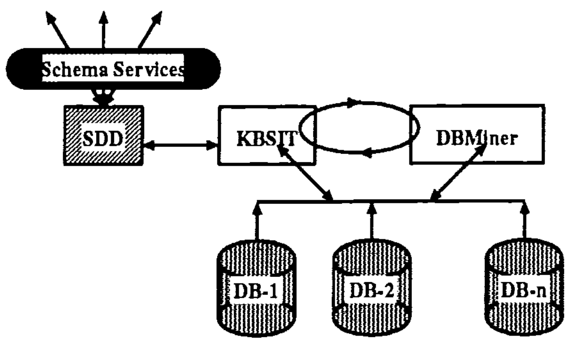

Figure 1: General Architecture

Applying data mining to schema integration (figure 1) is currently under implementation and experimentation in our Heterogeneous DataBase (HDB) prototype (Dao & Ebeid 1992) at Hughes Research Laboratories. *SDD* is the smart data dictionary process that maintains and services a federated schema to our heterogeneous database environment (Dao, Keirsey, & et al. 1991). *KBSIT* is the knowledge-based schema integration tool that exists as one of the services the SDD controls to construct and evolve the federated schema. *DBMiner* is an implementation of the attribute-oriented induction process described in (Cai, Cercone, & Han 1991; Fu & Han 1994). Whenever KBSIT requires knowledge about the relationship between two or more attributes, it constructs a *mining request* that is sent to DBMiner. DBMiner mines the databases to satisfy the request and returns a set classification and/or characterization rules. KBSIT analyzes these rules, as described herein, and computes a suggested relationship for the attributes of interest.

Data Mining for Schema Integration

Attribute-oriented Induction

In (Cai, Cercone, & Han 1991), an attribute-oriented induction method to extract rules from relational databases is developed. Assuming conceptual hierarchies on instances of attributes, each attribute value of a tuple can be replaced by a higher level concept. In this way, a generalized relation with fewer tuples is produced. This process is repeated until some threshold is reached. The final relation is then transformed into a logical formula, or a rule, according to the correspondence between relational tuples and logical formulas (Gallaire, Minker, & Nicolas 1984). Two kinds of rules can be derived in this approach: characteristic rules and classification rules.

We use $h(B = b_j|\{A_i\})$ to represent the request to mine for characteristic rules for instance b_j of attribute B in relevance to attributes $\{A_i\}$. $h(B|\{A_i\})$ represents mining for rules for all instances of B. Similarly $l(B = b_j|\{A_i\})$ and $l(B|\{A_i\})$ are be used for classification rule requests.

Figure 2 outlines the general process performed by attribute-oriented induction for learning characteristic rules (see (Cai, Cercone, & Han 1991) for the complete algorithm), where: (1) An input relation is constructed. (2) The training data is selected from the input relation (to learn rules for all graduate students, we select out the *Category* attribute and keep all tuples). (3) The training data is recursively "compressed" by concept hierarchies on attribute instances. Each attribute instance can be replaced by a more abstract instance from the concept hierarchies, then duplicate tuples are removed from the training relation. (4) A final "generalized relation" is computed. Each tuple in the generalized relation represents a logic formula characterizing the tuples in the initial training data. We can translate the generalized relation into logic formula(s) about the characteristic nature of the training data on the learning criteria (i.e., characteristic rules for graduate student tuples):

$$\forall(t).Category(t) \in graduate \Rightarrow$$
$$(Birth\_Place(t) \in Canada \wedge GPA(t) \in excellent) \vee$$
$$(Major(t) \in science \wedge Birth\_Place(t) \in foreign \wedge$$
$$GPA(t) \in good)$$

A similar process is performed to learn classification rules from an input relation of positive and negative instances of an initial tuple class.

The generalized relation for characteristic rules covers all positive data and forms a *necessary condition* of the target class, where *condition*(t) is a formula on attribute values of tuple t:

$$target\_attribute(t) \in target\_class \Rightarrow condition(t)$$

The generalized relation for classification rules distinguishes the target class from the contrasting class(es). The learned rule forms a *sufficient condition* of the form:

$$target\_attribute(t) \in target\_class \Leftarrow condition(t)$$

Applicability to Schema Integration

Our goal is to apply the data mining techniques of (Cai, Cercone, & Han 1991) to the problems of schema integration in our heterogeneous database prototype. We show how both classification and characteristic rules can be used to guide and suggest relationships among attributes.

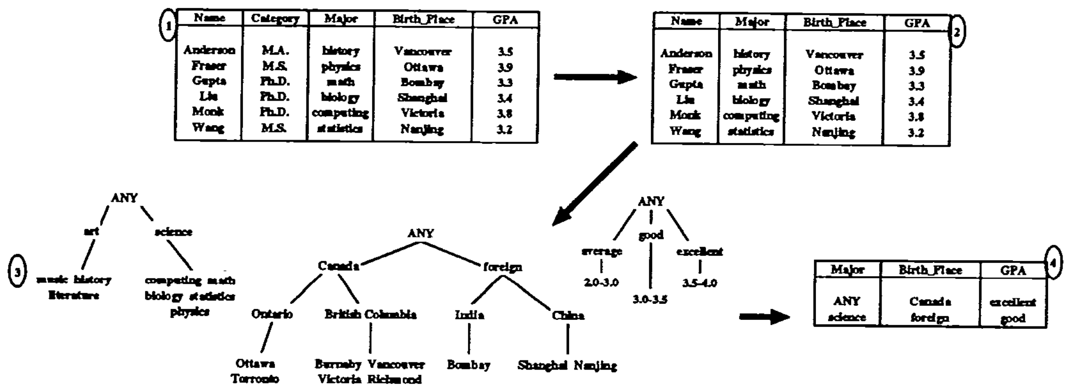

Figure 2: Attributed-oriented Induction Example

Applicability of Characteristic Rules For attribute B_1 from DB_1, we discover a set of characteristic rules CHR_1 (n is the number of instance values for B_1 in DB_1):

$$CHR_{11} : h(B_1 = b_{11}|\{A_i\}) \; \ldots \; CHR_{1n} : h(B_1 = b_{1n}|\{A_i\})$$

For attribute B_2 from DB_2, we discover a set of characteristic rules CHR_2 (m is the number of instance values for B_2 in DB_2):

$$CHR_{21} : h(B_2 = b_{21}|\{A_i\}) \; \ldots \; CHR_{2m} : h(B_2 = b_{2m}|\{A_i\})$$

Any tuple violating rule CHR_{ij} *cannot* participate in the tuple class ($B_i = b_{ij}$). Yet, if a tuple does satisfy CHR_{ij} then we have no information on its participation in tuple class ($B_i = b_{ij}$). To ascertain whether B_1 and B_2 represent similar attribute semantics, we compare the rules from CHR_1 and CHR_2. This is possible because, although B_1 and B_2 have unknown relationship, we constructed the two characteristic rule sets based on a set of attributes ($\{A_i\}$) known to be common to the two databases. Comparing a rule CHR_{1i} with a rule CHR_{2j} we find:

chr-1: $CHR_{1i} \cap CHR_{2j} = \emptyset$ (no tuple can satisfy both CHR_{1i} and CHR_{2j}): we know that b_{1i} and b_{2j} cannot be two encodings of the semantically compatible (equivalent, superclass, subclass) underlying instance (regardless of whether $b_{1i} = b_{2j}$).

Furthermore, if

$$\forall(i,j) \quad CHR_{1i} \cap CHR_{2j} = \emptyset$$

then we can assume that B_1 is semantically incompatible with B_2.

Applicability of Classification Rules For attribute B_1 from DB_1, we discover a set of classification rules CLR_1:

$$CLR_{11} : l(B_1 = b_{11}|\{A_i\}) \; \ldots \; CLR_{1n} : l(B_1 = b_{1n}|\{A_i\})$$

For attribute B_2 from DB_2, we discover a set of classification rules CLR_2:

$$CLR_{21} : l(B_2 = b_{21}|\{A_i\}) \; \ldots \; CLR_{2m} : l(B_2 = b_{2m}|\{A_i\})$$

Any tuple satisfying rule CLR_{ij} *must* participate in the tuple class ($B_i = b_{ij}$). Yet, if a tuple does not satisfy CLR_{ij} then we have no information on its participation in tuple class ($B_i = b_{ij}$). To ascertain whether B_1 and B_2 are representing similar attribute semantics, we compare the rules from CLR_1 and CLR_2. Comparing a rule CLR_{1i} with a rule CLR_{2j} we find:

clr-1: $CLR_{1i} = CLR_{2j}$ (every tuple satisfying CLR_{1i} will also satisfy CLR_{2j} and vice versa):

(1) if $b_{1i} = b_{2j}$ then these two rules suggest that B_1 is equivalent to B_2.

(2) if $b_{1i} \neq b_{2j}$ then B_1 is equivalent to B_2 only if there is a translation between instances b_{1i} and b_{2j} establishing equivalent semantics using different encodings.

clr-2: $CLR_{1i} < CLR_{2j}$ (every tuple satisfying CLR_{1i} also satisfies CLR_{2j}, but not vice versa):

(1) if $b_{1i} = b_{2j}$ then these two rules suggest that B_1 is a subclass of B_2.

(2) if $b_{1i} \neq b_{2j}$ then B_1 is a subclass of B_2 only if there is a translation M between b_{1i} and b_{2j} such that $M(b_{1i}) \leq M(b_{2j})$ (i.e., the translation of b_{2j} is equivalent to or a superclass of the translation of b_{1i}).

clr-3: $CLR_{1i} > CLR_{2j}$ (every tuple satisfying CLR_{2j} also satisfies CLR_{1i}, but not vice versa): this is the dual case of **clr-2**.

clr-4: $CLR_{1i} \cap CLR_{2j} \neq \emptyset$ (there exists tuples satisfying a subset of the conditions of CLR_{1i} and CLR_{2j} but not completely satisfying both): This "weak" correspondence suggests that B_1 and B_2 may be sibling concepts of some common, yet unknown, superconcept. We use the term "weak" because this correspondence provides nothing more than a casual suggestion that must be further verified before integration could take place using it.

clr-5: $CLR_{1i} \cap CLR_{2j} = \emptyset$ & $b_{1i} = b_{2j}$ (there is no correspondence between the classification rules on equal instance values b_{1i} and b_{2j}): B_1 can be in a positive relationship (equivalent, subclass, superclass) with B_2 only if there is a translation for the instances of B_1 and B_2 into a compatible encoding.

Deriving the Suggested Relationship We need to combine the comparisons of rules in CHR into a single integration conclusion. We form a table $chrTBL$ of dimension $n \times m$ where entry $chrTBL(i, j)$ represents the outcome from comparing rules CHR_{1i} and CHR_{2j}. Finally, a series of reductions/manipulations are performed to generate a single integration suggestion, termed $H(B_1/B_2|\{A_i\})$. $H(B_1/B_2|\{A_i\})$ will be one of $(inc, eq\_inc, null, \emptyset)$, where: inc says (B_1 incompatible with B_2); eq_inc says (B_1 and B_2 do not use equivalent encodings to represent attribute instances); $null$ says no consistent reduction exists; and $\emptyset$ says that $\{A_i\}$ did not provide enough relevance to mine attributes B_1 and B_2.

The reduction of $chrTBL$ to $H(B_1/B_2|\{A_i\})$ is actually quite straight-forward. If every entry in the table is $\emptyset$ (i.e., upholds relationship **chr-1**), then we reduce to conclusion inc. If every entry on the diagonal of $chrTBL$ is $\emptyset$, then we reduce to conclusion eq_inc. If some elements of $chrTBL$ are $\emptyset$ and others are not, then we reduce to $null$. Otherwise, there was not enough information to draw a conclusion about the two attributes.

Similarly we combine the comparisons of rules in CLR into a table $clrTBL$ and then a single integration conclusion, termed $L(B_1/B_2|\{A_i\})$. $L(B_1/B_2|\{A_i\})$ will be one of $(eq, sup, sub, sibl, null, \emptyset)$, where: eq says (B_1 equivalent with B_2); sup says (B_1 a superclass of B_2); sub says (B_1 a subclass of B_2); $sibl$ says (B_1 a sibling with B_2); $null$ says no consistent reduction exists; and $\emptyset$ says that $\{A_i\}$ did not provide enough relevance to mine attributes B_1 and B_2. The reduction of $clrTBL$ is more involved than that for $chrTBL$; but we perform the following basic algorithm[1]: (1) if outcome = (i.e., clr-1) exists at least once in every row and column of $clrTBL$, then we suggest eq as the relationship; (2) if outcomes = or $<$ occur at least once in every row and column, then we suggest *subclass* as the relationship; (3) if outcomes = or $>$ occur at least once in every row column, then we suggest *superclass*; (4) if both (2) and (3) hold, then we suggest $null$; (5) if outcomes =, $sibl$, $<$, or $>$ occur at least once in every row and column, then we suggest *sibling*; and (6) if none of (1)-(5) hold, we suggest $\emptyset$ as the relationship. If we are only concerned about the discovering the relationship when B_1/B_2 are using the same instance encodings, then the above reduction still holds but the "every row and column" must be the diagonal elements of $clrTBL$.

[1]The full paper (Dao & Perry 1995) contains details on the reduction of $chrTBL$ and $clrTBL$ to a single suggestion.

Data Mining Summary

Attribute-oriented induction has been presented as a tool to discover attribute relationships for schema integration. For our scenario, the integration engine is the application using attributed-oriented induction (e.g., requesting discovery knowledge from DBMiner) and responsible for generating mining requests ($h(B|\{A_i\})$ or $l(B|\{A_i\})$). KBSIT instructs DBMiner to learn classification and characteristic rules for data instances of two attributes from heterogeneous databases. The rules can then be compared to arrive at a singular suggestion, $H(B_1/B_2|\{A_i\})$ or $L(B_1/B_2|\{A_i\})$, for the integration relationship between attributes from two databases.

Schema Integration with Data Mining

KBSIT operates as an autonomous server to integrate relational schemas based on correspondences between attributes from the heterogeneous schemas. At any point during its processing, KBSIT will have: (1) a knowledge base of attribute relationships (i.e., correspondences) identified between two or more database schemas; and (2) an integrated representation of those relations rendered comparable by the knowledge base. When KBSIT reaches an impasse, where it cannot determine the relationship between two heterogeneous attributes, it must use its current knowledge base to derive data mining requests for DBMiner. From these requests, KBSIT can reduce the discovered rules to a common suggestion about semantic correspondence. Figure 3 summarizes this information flow between KBSIT and DBMiner.

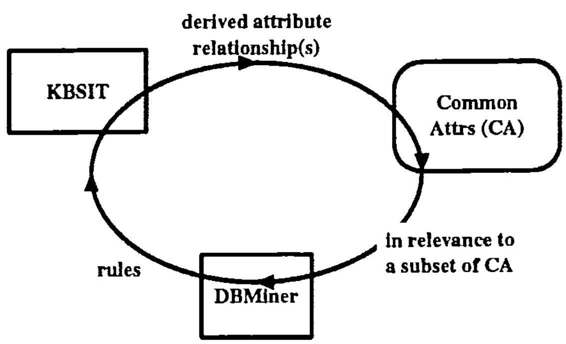

Figure 3: Schema Integration/Data Mining Loop

Schema Integration State

In order to understand how KBSIT generates requests to DBMiner, we will examine a "snapshot" of its integration processing. At any given point, KBSIT has a current set of federated attributes that represent those local database attributes that have been unified into common semantic structures (label this set CA). Now, should KBSIT be at a point where it has insufficient knowledge to unify attributes B_1 (from DB_1) and B_2

(from DB_2), it must generate a mining request based on CA to learn about B_1/B_2. In the naive case, we submit the request $L(B_1/B_2|CA)$ to learn based on all the common attributes we have identified[2]. Unfortunately, it may be the case that a subset of the attributes from CA properly determine the relationship between B_1/B_2; yet, when conditioned on all data from CA, DBMiner becomes inundated with meaningless data correlations and derives incorrect, null, or over-qualified rules. Thus, the data that B_1/B_2 is considered *in relevance to* is critical in the successful discovery of proper integration relationships.

A more positive way to examine the state of KBSIT is to consider the power set of CA and the 2^n subsets it represents. Ideally, we would like submit requests $L(B_1/B_2|S)$ for each set S in the power set of CA and gather, from DBMiner, the B_1/B_2 relationships for every possible relevance subset. Our task would then be to determine which relationship (from the power set requests) to use as the suggestion for B_1/B_2. We reduce to a single suggestion adhering to the following prejudices:

- We favor the discovery of a strong relationship over weaker relationships. If the data miner is able to recover *eq* as an implicit attribute relationship based on some relevance set, then we are inclined to use it over other suggestions (similarly for *sup* or *sub* over *sibl*).
- We favor few attribute correlations over complex, many attribute suggestions. This prejudice taints the mining process to prefer suggestions that are derived from fewer relevant attributes (i.e., smaller relevance clauses) – we use this assumption to guide the mining/integration algorithm to seek fundamental/primary relationships over spurious and artifactual coincidences.

DBMiner Control Process

Computing mining requests based on the power set of CA is, of course, not computationally practical. We must devise a control process in KBSIT that generates requests to DBMiner that approaches the ideal scenario.

We define an ordering and comparison function on the relationships computed for any two attributes B_1/B_2 as:

$$eq > sup > sibl \quad eq > sub > sibl$$

$$sibl > null > \emptyset \quad sup \neq sub$$

$$compare(r_1, r_2) = positive \ if \ r_1 \geq r_2, \ else \ negative$$

We begin mining for a relationships between B_1/B_2 by computing $L(B_1/B_2|\{A_i\})$ for each common attribute $A_i \in CA$. Then, for a specific $L(B_1/B_2|\{A_j\})$, we compute $L(B_1/B_2|\{A_j, A_k\}).k \neq j$ for all $k \in \{1 \ldots |CA|\}$. Any particular $L(B_1/B_2|\{A_j, A_k\})$ that does not *compare* as *positive* with $L(B_1/B_2|\{A_j\})$ marks the end of the "search" down this branch. For all positive steps, we then make a next step from $L(B_1/B_2|\{A_j, A_k\})$ to $L(B_1/B_2|\{A_j, A_k, A_p\}).j \neq k \neq p$ and apply the same "positive comparison" criteria for stopping or continuing search down this branch. Once we have exhausted the search from A_j, we compute the overall suggested relationship to be the most positive relationship in the search tree. This process is repeated beginning from every attribute $A_i \in CA$. The final suggested relationship for B_1/B_2 is then the most positive relationship among the set of A_i search trees. KBSIT can add this relationship to its knowledge-base and attributes B_1/B_2 to its set of common attributes.

Note that in the worst case this search is still the power set evaluation; but, in our experiments, we have found that the search tree is terminated fairly quickly and relationships are determined after minimal search iterations. Nevertheless, the control loop presented should be augmented to work in a breadth first fashion and/or perform a cutoff when the relevance sets grow too large.

[2]This section holds equally for $H(B_1/B_2|CA)$, we use only $L(B_1/B_2|CA)$ to minimize confusion and assume the parallel reading with H substituted for L.

Integration Analysis

In this section we summarize the analyses and lessons learned from applying the techniques presented herein to actual heterogeneous database environments. There are four general analyses that should be presented to judge the scope, applicability, and future directions of our current data mining for schema integration.

Analysis 1: The current algorithms use instance-level mining and comparison for relationship discovery. We have found that this process performs well for finite and discrete domains (i.e., *Birth_Place* or *Diagnosed_Disease*). Yet, the algorithms are too focused on individual instances to capture general relationships in continuous valued domains. We feel that intelligent pre-clustering techniques, applied to the attributes in isolation, may generate a discretized domain readily applicable to our mining/integration techniques.

Analysis 2: Attributes that require instance-level translations to unify are not satisfactorily handled. Our mining and rule-composition algorithms properly identify the need for translations for relationships to occur. Yet, how to suggest such translations is often extremely domain-specific. We have some initial techniques that use the knowledge mined to suggest translations, but this area needs more work to be acceptable in general schema integration servers.

Analysis 3: We currently compare all conditions from the mined rules to suggest relationships. We are working on techniques to use only those conditions that represent a high degree of coverage of the data instances. The motivation being that if one database has the only occurrences of an infrequently occurring attribute value, we do not want this value to adversely affect

or distort the mining results. The attribute-oriented induction process we are using to discover rules can compute this coverage for us, we need to define how to exploit it in the rule comparison/reduction algorithms.
Analysis 4: We currently generate and control mining requests to DBMiner in a heuristic manner that is non-tractable in degenerate cases. Clearly we need to incorporate algorithms such as those developed in (Agrawal & Srikant 1994) to control DBMiner in a tractable and optimized manner for all cases. Nevertheless, our heuristic control algorithm has behaved satisfactorily for a number of heterogeneous database case studies.

None of the above analyses identify flaws in the approach described herein. Instead, each outlines a task, or direction, required to extend the approach to a more general-purpose schema integration engine.

Conclusion

Our schema integration/data mining prototype has been tested on actual heterogeneous databases and the results prove promising that our approach leads to automated discovery of integration relationships based on multi-database correspondences (Dao & Perry 1995).

In this paper we discussed the problems and partial solutions to the difficult area of semantic-level heterogeneous schema integration. Specifically, we addressed the following points:

- How to *compare and combine knowledge* discovered by attributed-oriented inductive mining (Cai, Cercone, & Han 1991), applied to multiple databases, to gain insight on the semantic correspondence of attributes from heterogeneous schemas.
- How to generate, in a controlled and heuristic manner, the *mining in relevance to* clauses from the current set of common attributes.
- How to *combine the pieces* into a unified and controlled schema integration with data mining algorithm.
- The architecture of our *prototype system*. The ideas and algorithms described herein have been validated in our prototype system (figure 1).

We have outlined a new field for applying data mining technology and enhancing the difficult problem of schema generation/maintenance in federated database systems. We are currently experimenting extensively with the attribute-oriented data mining technique, but acknowledge that it will most likely take a parallel mining effort of various techniques to attain the level of semi-automated schema integration we seek.

Our future plans include completing, enhancing, and further testing the prototype system. We have only begun to see the results of mining for integration relationships and need to do more experimentation to validate our initial results. For the long-term, we have two goals in this new technology direction. First, we are building our prototype such that is not dependent on any one data mining technique, but uses a set of available techniques as services to aid the integration process. Second, we plan to investigate methods to exploit the instance-level constraints generated by data miners, coupled with domain knowledge, to suggest translations (i.e., scaling, mapping, etc.) between attributes for specific integration relationships to occur.

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. 20th VLDB.*

Batini, C.; Lenzerini, M.; and Navathe, S. 1986. A comparative analysis of methodologies for database schema integration. *Computing Surveys* 18(4).

Cai, Y.; Cercone, N.; and Han, J. 1991. Attribute-oriented induction in relational databases. In Piatetsky-Shapiro, G., and Frawley, W., eds., *Knowledge Discovery in Databases.* AAAI Press. 213–228.

Dao, S., and Ebeid, N. 1992. Interoperability of heterogeneous information management systems: a federated approach. Technical Report 585, Hughes Research Laboratories.

Dao, S., and Perry, B. 1995. Data mining for semantic schema integration: Extended report. Technical report, Hughes Research Laboratories.

Dao, S.; Keirsey, D.; and et al., R. W. 1991. Smart data dictionary: a knowledge-object-oriented approach for interoperability of heterogeneous information management systems. In *Proc. 1st International Workshop on Interoperability in Multidatabase Systems.*

Drew, P.; King, R.; McLeod, D.; Rusinkiewicz, M.; and Silberschatz, A. 1993. Report of the workshop on semantic heterogeneity and interoperation in multidatabase systems. *SIGMOD Record* 22(3).

Fu, Y., and Han, J. 1994. DBMiner user's guide. Technical report, School of Computing Science, Simon Fraser University.

Gallaire, H.; Minker, J.; and Nicolas, J. 1984. Logic and databases: A deductive approach. *ACM Computing Survey* 16(2).

Hayne, S., and Ram, S. 1990. Multiuser view integration system (muvis): An expert system for view integration. In *Proc. 6th International Conference on Data Engineering.*

Larson, J.; Navathe, S.; and Elmasri, R. 1989. A theory of attribute equivalence in database with application to schema integration. *IEEE Transactions on Software Engineering* 15(4).

Li, W., and Clifton, C. 1994. Semantic integration in heterogeneous databases using neural network. In *Proc. 20th VLDB.*

Navathe, S., and Buneman, P. 1986. Integrating user views in database design. *Computers* 19(1).

Piatetsky-Shapiro, G., and Frawley, W. 1991. *Knowledge discovery in databases.* AAAI Press.