REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-01-0188*

1. REPORT DATE *(DD-MM-YYYY)* 2002	**2. REPORT TYPE** Book Chapter	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Knowledge Discovery in Heterogeneous Environments		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHORS Magdi N. Kamel Marion G. Ceruti, Ph.D.		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SSC San Diego 53560 Hull Street San Diego, CA 92152–5001		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This is the work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction. Many SSC San Diego public release documents are available in electronic format at: http://www.spawar.navy.mil/sti/publications/pubs/index.html

14. ABSTRACT
This chapter addresses the topic of knowledge discovery in heterogeneous environments. It begins with an overview of the knowledge-discovery process. Because of the importance of using clean, consistent data in the knowledge-discovery process, the chapter focuses on the problems of data integration and cleansing by presenting a framework of semantic conflicts identification and an algorithm for their resolution. The chapter then describes the various data mining tasks that can be performed on the cleansed data, such as association rules, sequential patterns, classification and clustering. It also discusses data mining models and algorithms, such as those related to neural networks, rule induction, decision trees, K-nearest neighbors, and genetic algorithms. The chapter concludes with a summary.

Published in *Heterogeneous Information Exchange and Organizational Hubs*. Edited by H. Bestougeff, J. E. Dubois, and B. Thuraisingham. The Netherlands: Kluwer Academic Publishers, 2002.

15. SUBJECT TERMS

knowledge discovery	data mining
semantic conflicts identification	data mining tasks
semantic conflicts resolution	data mining algorithms

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Marion G. Ceruti, Ph.D., Code 246206
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19B. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	24	(619) 553–4068

2002 book chapter
Cerute

Heterogeneous Information Exchange and Organizational Hubs

edited by

H. Bestougeff

Professor,
University of Marne-la-Vallée, France

J.E. Dubois

Professor
University Paris VII, France

and

B. Thuraisingham

Chief Engineer,
Mitre Corporation, U.S.A.

20060425018

KLUWER ACADEMIC PUBLISHERS
DORDRECHT / BOSTON / LONDON

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 1-4020-0649-7

Published by Kluwer Academic Publishers,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Sold and distributed in North, Central and South America
by Kluwer Academic Publishers,
101 Philip Drive, Norwell, MA 02061, U.S.A.

In all other countries, sold and distributed
by Kluwer Academic Publishers,
P.O. Box 322, 3300 AH Dordrecht, The Netherlands.

Printed on acid-free paper

8. Building Secure Data Warehouse Schemas from Federated Information Systems
Fèlix Saltor, Marta Oliva, Alberto Abelló, José Samos 123
9. Conceptual MetaCube - Metadata for Multidimensional Cube
Nguyen Thanh Binh 135
10. Integration of Keyword-Based Source Search and Structure-Based Information Retrieval
Gengo Suzuki, Yuichi Iizuka, Shiro Kasuga 149

Part III Sharing Information and Knowledge Discovery 159

11. Knowledge Management for Heterogeneous Information Exchange
Bhavani Thuraisingham, Amar Gupta, Elisa Bertino, Elena Ferrari 159
12. The Exchange of Data in the Global Market
How to Avoid a Tower of Babel
Anna Moreno 169
13. Knowledge Discovery in Heterogeneous Environments
Magdi N. Kamel, Marion G. Ceruti 181
14. Heterogeneous Information Exchanges in the Construction Industry
The SYDOX/MATCOMP/Xi Prototype
Anne-Françoise Cutting-Decelle, Jacques-Emile Dubois 205
15. Databases and Design: Discovering New Ideas in Old Facts
Shuichi Iwata 223

Index 241

Chapter 13

Knowledge Discovery in Heterogeneous Environments

Magdi N. Kamel[1], Marion G. Ceruti[2]
[1]Naval Postgraduate School, [2]Space and Naval Warfare Systems Center, San Diego
Email: mnkamel@nps.navy.mil

Abstract: This chapter addresses the topic of knowledge discovery in heterogeneous environments. It begins with an overview of the knowledge-discovery process. Because of the importance of using clean, consistent data in the knowledge-discovery process, the chapter focuses on the problems of data integration and cleansing by presenting a framework of semantic conflicts identification and an algorithm for their resolution. The chapter then describes the various data-mining tasks that can be performed on the cleansed data, such as association rules, sequential patterns, classification and clustering. It also discusses data-mining models and algorithms, such as those related to neural networks, rule induction, decision trees, K-nearest neighbors, and genetic algorithms. The chapter concludes with a summary.

Key words: Knowledge discovery, semantic conflicts identification, semantic conflicts resolution, data mining, data mining tasks, data mining algorithms

1. INTRODUCTION

The explosive growth of government, business, and scientific databases has overwhelmed the traditional, manual approaches to data analysis and created a need for a new generation of techniques and tools for intelligent and automated knowledge discovery in data. The field of knowledge discovery is an emerging and rapidly evolving field that draws from other established disciplines such as databases, applied statistics, visualization, artificial intelligence and pattern recognition that specifically focus on fulfilling this need. The goal of knowledge discovery is to develop techniques for identifying novel and potentially useful patterns in large data sets. These identified patterns typically are used to accomplish the following goals [9]:

- to make predictions about new data,
- to explain existing data
- to summarize existing data from large databases to facilitate decision making, and
- to visualize complex data sets.

Knowledge discovery is as an interactive and iterative process that consists of a number of activities for discovering useful knowledge. The core activity in this process is data mining, which features the application of a wide variety of models and algorithms to discover useful patterns in the data. Whereas most research in knowledge discovery has concentrated on data mining, other activities are as important for the successful application of knowledge discovery. These include data selection, data preparation, data cleaning, and data integration. Additional activities are required to ensure that useful knowledge is derived from the data after data-mining algorithms are applied. One such activity is the proper interpretation of the results of data mining.

Whereas most efforts have been on knowledge discovery in structured databases, there is an increasing interest in the analysis of very large document collections and the extraction of useful knowledge from text. More than 80% of stored information is contained in textual format, which cannot be handled by the existing knowledge discovery and data mining tools. With the popularity of World Wide Web, the vastness of the on-line accessible information has opened up a host of new opportunities. At the same time, it accelerates the need for new tools and technologies for textual-content management and information mining.

2. THE KNOWLEDGE-DISCOVERY PROCESS

In its simplest form, the process of knowledge discovery is an iteration over three distinct phases: data preparation, data mining, and presentation. The first phase, data preparation, can be further broken down into two steps: 1) data integration and cleansing and 2) data selection and preparation. Data integration is the process of merging data from multiple heterogeneous sources or databases. Data cleansing is a collection of processes that aim at improving the quality of the data. These processes may include the removal of duplicate data, handling missing values, identifying and resolving semantic ambiguities, cleaning dirty sets, etc. Because integration and cleansing issues are common with those found while building a data warehouse, having a data warehouse in place greatly facilitates the knowledge-discovery process. However, data mining does not require a data warehouse to be built, rather select data can be downloaded from operational databases, integrated, cleansed and input to a data-mining processor.

Data selection and preparation involve identifying the data required for mining and eliminating bias in the data. Identifying the data relevant to a given mining operation is a difficult task that requires intimate familiarity with the application-domain data and the mining operation being performed on those data. Identifying the bias in the data is an equally important step, since bias in data can lead to the discovery of erroneous knowledge. Therefore, bias in data should be detected and removed prior to performing any data-mining operations.

Data mining actually occurs in the second phase of the knowledge-discovery process. Integrated and cleansed data generated in the first phase are accessed by the data-mining processor directly or through a middleware.

The third phase of the knowledge-discovery process is the presentation of facts discovered during the data-mining process. The presentation can be done by the data-mining processor or by a separate presentation front-end tool.

In the remainder of this chapter we address data integration and cleansing issues and the data mining process itself.

3. DATA INTEGRATION AND CLEANSING IN HETEROGENEOUS ENVIRONMENTS

As discussed in the previous section, a crucial step for knowledge discovery is the integration and cleansing of data from heterogeneous sources. Integrating data in this environment involves resolving the heterogeneity on three main levels, namely, the platform, data model (sometimes termed "syntactic") and semantic levels. (See, for example [4], [5], [6].) Platform integration addresses the coarsest level of heterogeneity, such as database management system (DBMS) vendors, DBMS transaction processing algorithms, and DBMS query processing [4].

Data integration involves removing inconsistencies that arise from different schemata or data models, different DBMS query languages and versions, different integrity constraints, etc. [4]. Detecting and resolving errors on this level is more complicated than that of the platform level because it involves merging business-process models and policies that may conflict, assuming that they are all known. For example, a data administrator in organization A may require that employees must enter their names, social-security numbers, full addresses, phone numbers, and vehicle license-plate number as "not-null" attributes. In contrast, a data administrator from organization B may require as mandatory only the name, social-security number, phone number, and address but allow the home phone-number and vehicle-license-number attributes to be null. When the organizations merge, the employee databases also would need to merge. Consequently, in the absence of any additional policy guidance, the database administrator must choose whether to relax the not-null constraint for the attributes in the employee database of organization A, or to require the employees of organization B to provide their home phone numbers and vehicle license numbers.

In next two sections we address the issue of identifying and resolving semantic conflicts of data collected from multiple heterogeneous sources.

4. SEMANTIC CONFLICTS IN HETEROGENEOUS ENVIRONMENTS

The most challenging level at which to integrate data is the semantic level. (See, for example [4], [5], [6].) To be sure that a data set contains all semantically consistent data is a task that is NP complete, due to the combinatorial explosion. That is, as the number, N, of data elements approaches infinity, the number of two-way comparisons rises faster than a polynomial in N. Given that this is the case, one must look toward heuristics to simplify that task into the realm of tractability. (See, for example, [8]).

Semantic heterogeneity is an adverse condition characterized by disagreement about the meaning, interpretation, or intended use of the same or related data [13]. Semantic heterogeneity can be classified broadly into two categories, schema and data.

Schema conflicts include homonyms and synonyms, as well as differences in data types, length, units of measure, and levels of object abstraction. Schema conflicts such, as homonyms and synonyms, can be determined at schema-definition time. Other schema conflicts, including differences in data domains and units of measure, also can be determined at schema-definition time, provided this information is specified as part of the attribute name [6]. In other cases, it can be obtained from data dictionaries.

Data conflicts are best discovered and can be verified only at run time using queries against various database components. Data-fill heterogeneity includes different units of measure, different levels of precision, different data values for the same measurement, and different levels of granularity. The heterogeneity classification process can include three distinct, but related levels of semantic heterogeneity [6]. This classification contributes to a logical progression to simplify and facilitate the development of the algorithm described in Section 5.

Levels of abstraction or granularity pertain to objects and also to the information describing them. For example, levels of object abstraction in semantic heterogeneity pertain to physical or notional entities, such as a fleet of ships (coarse), and individual ships (fine). Categories of semantic heterogeneity can be arranged conveniently according to information granularity. For example, conflicts in the data category arise from differences in data values returned by similar queries against different databases with attributes representing the same objects. Kamel has described a detailed classification of semantic conflicts with examples from Naval Administrative Databases [10].

4.1 Level-One Granularity – Relations

Relations are database-structural components at the most coarse-grained level of information. This level is limited to names and definitions of relations, both in comparison to the names and definitions of other relations as well as in comparison to those of attributes. The resolution of semantic inconsistencies at level one does not require access to the data fill. Relation-attribute homonyms occur when a relation and an attribute have the same name. To avoid ambiguity, all relations and attributes should have unique names, but this is not always the case when merging data sets that originated from different organizations into the same data warehouse.

4.2 Level-Two Granularity – Attributes

The attribute level of granularity includes data-element names, definitions, meanings, data types and lengths. For example, a synonym occurs when the same entity is named differently in different databases, e.g. "vessel" in database A and "ship" in database B. Analysis at level two can resolve semantic conflicts to produce unique attribute names, definitions, data types and lengths for entities in a global, integrated schema. Attributes with different representations in the local schemata that have the same representation in the global schema after analysis at level two can be called "equivalent attributes" [6].

Data-type conflicts occur when equivalent attributes have different data types (e. g., character vs. numeric), particularly when integrating data managed in different Database Management Systems (DBMSs). For example, one DBMS may use an integer type, whereas another may use a numeric type for the same purpose. Similarly, length conflicts occur when equivalent attributes have different lengths. Type conflicts are quite common when dealing with databases designed for different implementations, whereas length and range conflicts are more likely to occur as a result of semantic choices [10]. The risk of synonyms increases if two users adopt vocabularies at different abstraction levels [2].

A homonym occurs when different objects or concepts (e.g. entities and attributes) are assigned the same name in different component databases. The risk of homonyms generally is higher when the vocabulary of terms is small, whereas the risk of synonyms is higher when the vocabulary of terms is rich [2]. In a data warehouse that contains integrated databases, all instances of semantic heterogeneity at level one and most at level two can be discovered by analyzing the results of appropriate queries on the metadata, assuming the metadata are consistent with and correctly represent the database design. Homonym analysis at level two frequently can be performed without consulting the data fill. Detecting synonyms is more difficult because the wording of data definitions can vary whereas the meanings remain identical [6].

4.3 Level-Three Granularity – Data Fill

Level three has the finest granularity and is needed because many semantic conflicts cannot be resolved at the schema level due to incomplete specification of the metadata. Detection of semantic heterogeneity at level three requires access to the data fill to obtain a more precise specification of the domains of attributes that appear to be equivalent at level two in order to determine whether these attributes represent the same or different objects. Semantic conflicts in different domains at this level arise from different units of measure, different levels of precision, and different ranges of allowable values [6].

Conflict resolution at level three requires an understanding of domains, which are the sets of all allowed data-element values for attributes. The resolution of semantic inconsistencies at the data-fill level has been hampered by the complexity of domain issues. Frequently, the schema is not sufficiently explicit to specify the domains. Resolution of semantic inconsistencies at this level can be very difficult. Precise domain definitions are required for the complete resolution of semantic heterogeneity. Strong typing is one way to address this problem, but this can be time consuming and expensive [6].

Given this constraint, the most comprehensive solution at the data-fill level that is theoretically possible can be achieved only by considering data updates and data implementation, which are outside the scope of this chapter. Therefore, we offer a partial solution that depends on assumptions necessitated by the lack of strong typing. One such assumption is that a "select-distinct" query will represent the domain sufficiently for semantic-conflict resolution to occur [6].

5. SEMANTIC-CONFLICT-RESOLUTION ALGORITHMS

Alternatives are available to simplify the semantic-integration task. For example, one can focus on data that are used most often and also resolve conflicts in data that are of known critical operational importance [8]. Other pragmatic approaches include resolving conflicts that are easy to identify with a minimum of expended resources, or focusing on data in specific categories that have been known to exhibit semantic conflicts in data sets previously observed [8].

When the search domain has been restricted sufficiently, some algorithms become tractable to use on small- to moderate-sized data sets. For example, [6] contains a detailed algorithm that is designed to resolve some conflicts on the three levels described in Sections 3 and 4. This algorithm addresses the semantic level by systematically searching for the following kinds of semantic heterogeneity: synonyms, homonyms, conflicts in data lengths, types, and units.

5.1 Sample Data Set Description

The algorithm is explained below in terms of an example from the Global Command and Control System - Maritime, (GCCS-M), which is the result of a comprehensive systems-integration effort that supports the U. S. Navy, the Marine Corps and the Coast Guard. A significant contribution to GCCS-M and its predecessors comes from the Naval Warfare Tactical Database (NWTDB), which is the standard, authoritative data source for all Naval tactical warfare systems [4], [7]. Due to the diverse data sets of NWTDB, the database integration necessary to form NWTDB served as a model for the GCCS-M database integration which includes not only data from NWTDB but other databases required to support a wide variety of maritime Command, Control, Communications, Computers and Intelligence (C^4I) applications with diverse DBMSs. These database-integration efforts provided metadata for case studies in integrating data dictionaries and identifying semantic conflicts [4], [5], [6].

Table 1 presents sample metadata of some NWTDB components. Because the GCCS-M federated database (FDB) resulted from an integration of several different data sources, the GCCS-M data categories are represented explicitly in the NWTDB and also in Table 1. Component databases designated under "DB" represent NWTDB data sources: "GR" – GCCS-M FDB readiness data from GCCS-M ashore; "GT" – GCCS-M FDB track data from GCCS-M ashore; "M" – Modernized Integrated Database (MIDB) from GCCS-M afloat [6].

Table 1 shows an example of a Synonym-Homonym Group (SHG) which is a set of two or more attributes related by synonymy or homonymy or both [4], [5]. SHGs also can be called "semantically heterogeneous groups." The use of SHGs enables a clear focus on the common ground and the diversity among related component databases. The SHG in Table 1 includes both synonyms and homonyms. Other researchers also have used clustering techniques similar to SHGs to identify trends in data. (See, for example, [12]).

Attribute name	Relation name	Data Type	Data Length	DB*	Attribute Definition
HULL	ESS_MESSAGE_D_E	CHAR	6	GR	Hull number.
HULL	TRKID	CHAR	24	GT	Hull number of ship, submarine, squadron number for fixed-wing aircraft.
HULL_NUMBER	IDBUQL	CHAR	15	M	Hull number of a vessel.

DB* = Database; GT = GCCS-M Track Database; GR = GCCS-M Readiness Database; M = Modernized Integrated Database

Table 1. Examples of Synonym-Homonym Group Derived from C^4I Data Sets in the Naval Warfare Tactical Database [6]

5.2 Algorithm Features

In this section, we summarize an algorithm for identifying and resolving semantic heterogeneity using heuristics. (For a detailed description of the algorithm, see [6].) The objective of the algorithm is to construct a consistent, global, integrated schema for databases A and B that can facilitate data mining and knowledge discovery. Each phase of this algorithm is based on one of the levels of information granularity described above in Section 4. Connection points are also specified for navigation between levels so that transitions between levels can proceed in a logical manner when resolving conflicts. For example, when the inconsistencies are resolved at the relations level, attention then is directed toward the attributes level [6].

This algorithm can be generalized to apply to the schemata of any number of component databases in a data warehouse and is useful in identifying most, if not all, of the SHGs present in the aggregate of the component databases. This approach enables data from operational systems to be cleaned periodically and ported into an integrated data warehouse [8].

The algorithm was designed to identify and resolve a hierarchy of semantic conflicts, some of which can be resolved by data-dictionary comparison and some of which will require an analysis of the data fill and/or specific domain knowledge at schema-definition time. The algorithm features a systematic procedure designed to ensure that the analyst will not omit inadvertently the comparisons between relations, attributes, and data fill of the component databases [8].

The methodology was designed to resolve semantic inconsistencies at each level before progressing to the next lower level. One proceeds to the next level only when finished at the higher one or when information is needed from a lower level to complete the analysis at the higher one. The algorithm is intended to be applied recursively to the metadata until each instance of semantic heterogeneity is resolved. Thus, the algorithm can be applied to the entire metadata in case all SHGs are not identified, although SHG formation prior to algorithm usage facilitates efficiency by ignoring attributes without semantic inconsistencies. The methodology is designed to eliminate from further consideration metadata irrelevant to semantically related groups, such as SHGs [8].

5.3 Example of Algorithm Application

The following application of some of the heuristics in the algorithm illustrates a relatively simple example of the identification and resolution of semantic heterogeneity in the ship-identifier SHG in Table 1.

The algorithm can be applied to all metadata at the relations level to generate the SHGs by conducting pairwise comparisons between all relation names and attribute names in the three databases. (Some minor details of the procedure have been omitted for brevity in this example.)

The following heuristics were extracted from the algorithm. Each heuristic is followed by an observation concerning the result of its application [8].

- Compare the names of the relations in databases GR, GT and M. All relation names are unique.

• Compare names of relations to names of attributes. All three relation names differ from all three attribute names.

• Compare relation descriptions. Whereas examples of relation descriptions are not included in this paper, an analysis of the relation descriptions indicates that the relations were designed for unique purposes. Thus application of this heuristic reveals no semantic inconsistencies at the relations level.

• Continue analysis at the attribute level.

• Compare attribute names in database GR to those in databases GT and M, etc. HULL occurs in two of the three databases.

• Compare attribute definitions. HULL has different definitions in databases GR and GT; thus, they are homonyms.

• Compare meanings of attribute definitions. HULL in database GR has a definition equivalent to HULL_NUMBER in database M.

• Compare data-element types and lengths. The application of this heuristic reveals same data type, but different lengths. Thus, HULL in database GR and HULL_NUMBER are class-two synonyms.

• Continue analysis at the data-fill level.

• Compare domains of data fill for HULL-related attributes in all three databases. The domains for HULL and HULL_NUMBER are the same in databases GR and M, respectively. This domain is a subset of the domain for HULL in database GT.

• Return to the attribute level.

The semantic conflicts are identified using the information obtained at the data-fill level. Therefore, the algorithm returns to the attribute level to resolve the inconsistencies.

• Rename HULL in database schema GR and HULL_NUMBER in database schema M. The new attribute name is "HULL_VESSEL."

• Change the attribute definition in database GR to ***"Hull number of a vessel."***

• Increase the length of the "HULL_VESSEL" attribute in database schema GR from 6 to 15 characters.

Semantic heterogeneity is identified and resolved at the attribute level.

Table 2 shows the results of the algorithm's application in which the semantic heterogeneity in the ship-identifier SHG from Table 1 has been resolved. Table 2 displays the modified metadata in ***bold italics***.

Attribute name	Relation name	Data Type	Data Length	DB*	Attribute Definition
HULL_VESSEL	ESS_MESSAGE_D_E	CHAR	15	GR	***Hull number of a vessel.***
HULL	TRKID	CHAR	24	GT	***Hull number of ship or submarine, squadron number for fixed-wing aircraft.***
HULL_VESSEL	IDBUQL	CHAR	15	M	Hull number of a vessel.

DB* = Database; GT = GCCS-M Track Database; GR = GCCS-M Readiness Database;
M = Modernized Integrated Database.

Table 2. Processed Metadata from Table 1 with Semantic Conflicts Resolved [6]

5.4 Limitations of the Methodology

The methodology includes decisions about resolving semantic heterogeneity that are somewhat arbitrary because of the arbitrary nature in which many attribute and relation names and definitions are selected in autonomous, legacy databases. Moreover, the manner in which attributes and data fill are separated in the autonomous databases also is arbitrary.

The methodology depends on the assumption that an analyst can judge whether data entities are the same or different. Sometimes the context is ambiguous, particularly with class-two synonyms if they cannot be resolved at the data-fill level. Analysis at this level is the most difficult because knowledge of data updates and implementations may be required for the resolution of some data-type heterogeneity. For example, if an attribute requires a numerical data type, a format error could result from an update to the attribute if the allowed data-type requirement has been relaxed to the more general character data type.

This methodology covers several properties of relations, attributes and their data fill. Heterogeneity with respect to nullness; differences in levels of security; data updates; and some kinds of data granularity, except at the relations level, were ignored.

The methodology can report character-numerical domain mismatches, but it cannot resolve them without the input of a data analyst or the use of knowledge-based techniques. Similarly, heterogeneity due to different levels of precision can be discovered but not resolved at the data-fill level.

An implicit assumption during the implementation of this algorithm is that no updates or modifications of any aspect of the component databases will be allowed because these changes could interfere with conflict discovery and resolution.

Whereas this section is intended to describe a framework for the systematic resolution of semantic inconsistencies, more work is needed in this area, especially to address conflicts arising from data updates and intended use. Because of the variety and complexity of semantic problems, this methodology is appropriate for detecting and resolving some, but not all semantic inconsistencies.

Finally, the algorithm's performance is expected to degrade in the limit of large data warehouses. This is the reason for restricting the search-domains prior to the application of the algorithm. (See, for example [8].)

Certain costs and tradeoffs are associated with selecting the degree of restriction in the search domain. The more restricted the domain, the more efficient the integration task. However, many data sets will be excluded from consideration. A broader domain will lead to a more comprehensive integration result but the task will be much more complex, resource intensive, and time consuming.

As with any effort, practical consideration must be given to the resources available to perform the data integration and the resolution of semantic heterogeneity. Cost and schedule must be considered, especially when dealing with very large data sets that easily could occupy several database administrators for many months with integration-verification tasks. As in so many other areas, a tradeoff exists between cost, schedule and completeness of a data-integration task. For this reason, it is important to build into the schedule sufficient time for data quality control and integration of heterogeneous data sources.

Artificial intelligence techniques, such as the use of ontologies as aids to semantic integration also have proven useful. (See, for example [3]).

6. DATA MINING TASKS

Data are ready for mining when they have been selected, cleansed, and integrated. A data-mining task needs to be selected before data mining can occur. A data-mining task refers to the type of problem to be solved. In this chapter we discuss the four prominent types of problems to be solved by data mining: association, sequential patterns, classification, and clustering.

6.1 Association Rules

Association rules associate a particular conclusion (e.g., the purchase of Brand X) with a set of conditions (e.g., the purchase of several other brands). An association rule is presented in the form:

CONCLUSION ⇐ CONDITION 1 & CONDITION 2 & ...& CONDITION N

For example:

Beer ⇐ Cigarettes & Frozen Foods

This rule indicates that people who buy cigarettes and frozen meals are likely to buy beer too. Each rule is assigned two measures to indicate the strength of the association for each rule. These measures are called **coverage** (or **support**) and **accuracy** (or **confidence**). **Coverage** is the proportion of records in the data set that have the set of CONDITIONS occurring together. **Accuracy** is the proportion of those records that have the CONDITIONS and CONCLUSIONS. These measures are often represented in the following format:

CONCLUSION ⇐ CONDITIONS (Number of records: Coverage, Accuracy)

For example:

Beer ⇐ Cigarettes & Frozen Foods (2341: 10%. 0.8)

which indicates that 10% of the customers (out of 2341 individuals in the data set) bought cigarettes and frozen foods. Of these 10% (234 individuals), 80% also bought beer.

The aim of the algorithm that discovers associations is to find all association rules with coverage >= minimum coverage and accuracy >= minimum accuracy, where minimum coverage and minimum accuracy are specified by the user.

Discovery of associations is the process of finding answers to questions such as: When people buy brand X do they tend to buy brand Y? If people have high cholesterol do they tend to have high blood pressure?

Discovering associations in data sets is very useful in applications such as brand positioning, advertising, direct marketing, medical diagnosis, and in military battlefield planning.

6.2 Sequential Patterns

In a sequential pattern task, sets of related records collected over time are analyzed to detect frequently occurring chronological patterns. For example sequential patterns can be used to identify the set of purchases that frequently precede the purchase of a microwave oven. Another example could be the discovery of a pattern that 70% of the time when Stock X increased its value by at most 10% over a 5-day trading period and Stock Y increased its value between 10% and 20% during the same period, the value of Stock Z also increased in a subsequent week. Since records occurring at different times need to be related, this approach requires that individual records be tagged with the identity of the entity that they represent.

6.3 Classification

Given a set of records, each comprised of a number of predicting attributes and a goal attribute that classify each record depending on its value, the goal of a classification task is to discover the relationship between the predicting attributes and the goal attribute. The discovered relationship is used to classify new records of unknown classification by predicting the value of the goal attribute. The class description generated by a classification task may be expressed explicitly by a set of rules describing each class or implicitly using a mathematical function which derives the class to which a record belongs to when this record is given as input to this function.

The algorithms and techniques used for class description are commonly called "predictive modeling," since the inputs are used to predict the value of an output. Many classification models have been developed. They include linear regression models, decision-tree models, rule induction, and neural network models. Rule-induction classifiers are examples of explicit classifiers whereas neural-network classifiers are examples of implicit classifiers.

A well-suited application for the classification task is that of credit analysis. A bank or credit card company usually has records about its customers that include characteristics of these customers such as income, age, number of children, number of credit cards, etc. For those customers for which their credit history is known, their records also include an attribute that describes their credit risk (e.g., Good, Medium, or Poor). A predictive model can examine these records and produce an explicit or implicit description of these classes. For example, an implicit model would produce a rule that specifies a "Good" credit risk for those customers who earn more than $40,000, are between the ages of 40 and 50 and have no children.

Classification is used extensively in applications such as credit risk analysis, portfolio selection, health-risk analysis, image and speech recognition, etc. Classification is particularly important for image-identification application, such as those used in military intelligence activities.

6.4 Clustering

Unlike the classification task, the clustering task input records do not contain a goal attribute. No classes are known at the time the clustering model is applied. The goal of the clustering task is to produce a reasonable classification of the set of input records according to some criteria defined by the clustering model. In some respects, the clustering model 'invents' classes by grouping records with similar attribute values into the same class. Similar to classification models, clustering models may produce explicit or implicit descriptions of the resulting segmentation.

Example applications that can use clustering tasks are market segmentation, discovering affinity groups, and defect analysis.

It is important to note that different data-mining tasks can be used in a cooperative fashion. For example, an association task could be used to identify a group of products that are likely to be purchased together or a sequential pattern function can identify a group of customers that are likely to purchase a specific product after purchasing other products. A classification function can then be used to produce a generalized description of products or customers in this class. Similarly a clustering task can be used to classify a set of records according to some criteria. After clustering, classification methods can be applied to discover rules to predict membership in a given class.

7. DATA MINING MODELS AND ALGORITHMS

Whereas data-mining tasks refer to the type of problems to be solved, data-mining models and algorithms are the methods used to solve a particular data-mining task. Several combinations of data-mining tasks and data-mining algorithms are possible. This means that a data-mining task can be accomplished using several data-mining algorithms, and a data-mining algorithm could be used to complete several data-mining tasks.

7.1 Neural Networks

The neural-network approach is based on constructing computers with architectures and processing capabilities that attempt to mimic the architecture and processing of the human brain [11]. A neural network is a large network of simple processing elements (PEs) which process information dynamically in response to external inputs. The processing elements are simplified representation of brain neurons. The basic structure of a neural network consists of three layers: input, intermediate (called the *hidden layer)*, and output. Figure 1 depicts a simple three-layer network.

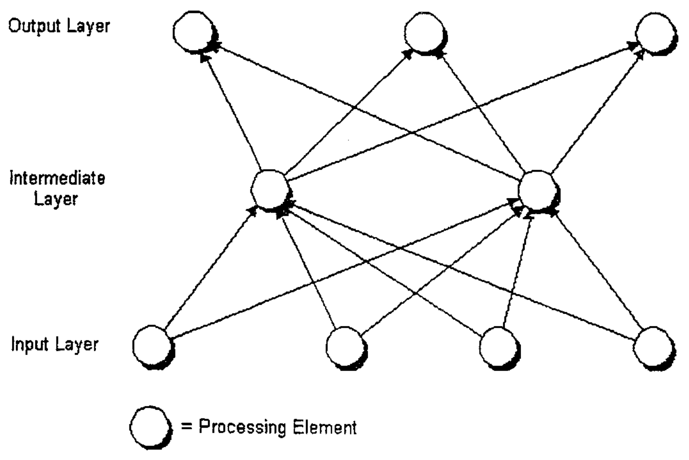


Figure 1. A three-Layer Neural Network Architecture. The Layers of the Network are the Input, Intermediate (Hidden), and Output Layers.

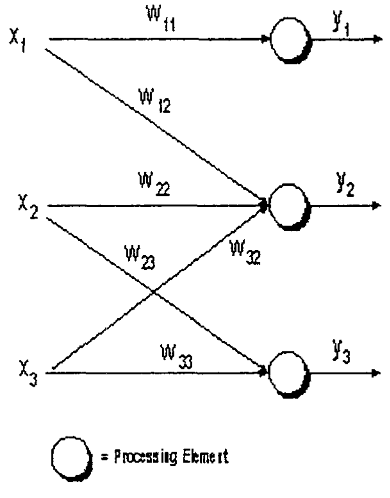
Each processing element corresponds to a predictor variable. It receives inputs, processes the inputs, and generates a single output. Each input corresponds to a decision factor.

For example, for a loan approval application, the predictor variables may be the income level, assets, or age. The output of the network is the solution to the problem. In the loan approval application a solution may be simply a "yes" or "no." A neural network, however, uses numerical values only to represent inputs and outputs.

Each input x_i is assigned a *weight* w_i that describes the relative strength of the input. Weights serve to increase or decrease the effects of the corresponding x_i input value. A summation function multiplies each input value x_i by its weight w_i and sums them together for a weighted sum y. As Figure 2 illustrates, for j processing elements, the formula for n input is:

$$y_j = \sum_j w_{ij} x_i$$

Based on the value of the summation function, a processing element may or may not produce an output. For example, if the sum is larger than a *threshold value* T, the processing element produces an output y. This value may then be input to other nodes for a final response from the network. If the total input is less than T, no output is produced. In more sophisticated models, the output will depend on a more complex activation function.



$$y_1 = x_1 w_{11}$$
$$y_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32}$$
$$y_3 = x_2 w_{23} + x_3 w_{33}$$

Figure 2. Summation Function for a Number of Neurons.

The knowledge in a neural network is distributed in the form of internode connections and weighted links. These weights must be learned in some way. The learning process can occur in one of two ways: *supervised* and *unsupervised learning.*

In supervised learning, the neural network is repeatedly presented with a set of inputs and a desired output response. The weights are then adjusted until the difference between the actual and desired response is zero. In one variation of this approach, the difference between the actual output and the desired output is used to calculate new adjusted weights.

In another variation, the system simply acknowledges for each input set whether or not the output is correct. The network adjusts weights in an attempt to achieve correct results. One of the simpler supervised learning algorithms uses the following formula to adjust the weights w_i:

$$w_i(new)=w_i(old)+\alpha*d*\frac{x_i}{|x_i|^2}$$

where α is a parameter that determines the rate of learning, and d is the difference between actual and desired outputs.

In unsupervised learning, the training set consists of input stimuli only. No desired output response is available to guide the system. The system must find the weights w_{ij} without the knowledge of a desired output response.

Neural networks can automatically acquire knowledge from historical data. In that respect they are similar to rule induction. They do not, however, need an initial set of decision factors or complete and unambiguous sets of data. Neural networks are particularly useful in identifying patterns and relationships that may be subsequently developed into rules for expert systems. Neural networks could also be used to supplement rules derived by other techniques.

7.2 Rule Induction

Rule induction is an inductive learning method in which rules are generated from example cases [11]. A rule-induction system is given an example set that contains the problem knowledge together with its outcome. The rule-induction system uses an induction algorithm to create rules that match the results given with the example set. The generated rules can then be used to evaluate new cases where the outcome is not known.

Consider the simple example set of Table 3, which is used in approving or disapproving loans for applicants. Application for a loan includes information about the applicant's income, assets, and age. These are the decision factors used to approve or disapprove a loan. The data in this table show several example cases, each with its final decision. From this simple example case, a rule-induction system may infer the following rules:

1. If income is high, approve the loan

2. If income is low, assets are high, approve the loan
3. If income is medium, assets are medium, and age is middle or higher, approve the loan

Name	Annual Income	Assets	Age	Loan Decision
Applicant A	High	None	Young	Yes
Applicant B	Medium	Medium	Middle	Yes
Applicant C	Low	High	Young	Yes
Applicant D	Low	None	Young	No

Table 3. Example Data Set from a Loan Application Database Used for Rule Induction

The heart of any induction systems is the induction algorithm that is used to induce rules from examples. Induction algorithms vary from traditional statistical methods to neural computing models.

Rule induction offers many advantages. Unlike decision trees, rule induction does not force splits at each decision level. Therefore, one can look ahead and perhaps find different, and sometimes better, patterns for classification.

Induction systems, however, suffer from several disadvantages. Unlike decision trees, the rules generated may not cover all possible situations. Also unlike trees, the rules generated may be conflicting, in which case a conflict resolution strategy is needed to choose which rule to follow. Further, they can generate rules that are difficult to understand. They may require a very large set of examples to generate useful rules. In some cases, the examples must be sanitized to remove exception cases. Additionally, the computing power required to perform the induction grows exponentially with the number of decision factors.

7.3 Decision Trees

Decision trees are a graphical representation of a problem domain search space. A decision tree is composed of nodes and branches. Initial and intermediate nodes represent decision attributes, and leaf nodes represent conclusions. A path from the root node to a leaf node leads to a class or value. Figure 3 shows a decision tree for investment decisions based on age, amount of investment, and investment style.

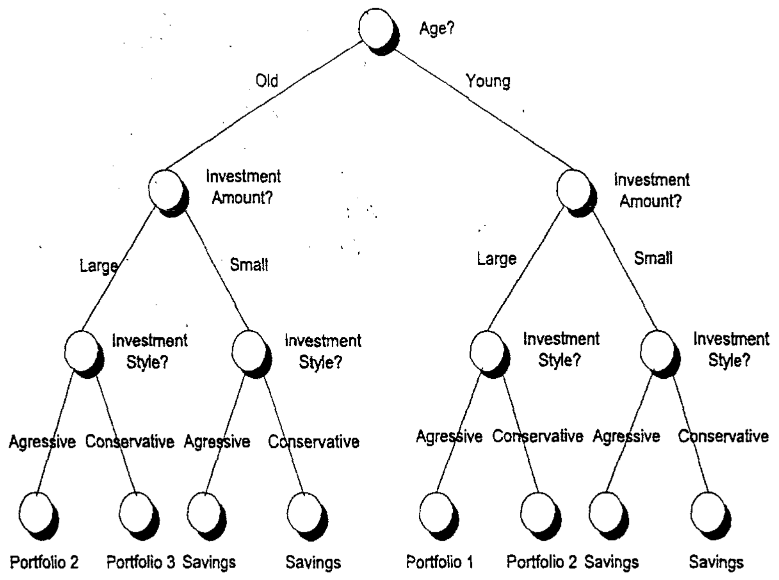


Figure 3. Example of a Decision Tree for the Investment Decisions.

Decision trees are commonly used as predictive models. Decision trees used to predict categorical variables are called *classification trees*, and those used to predict continuous variables are called *regression trees*. Different types of algorithms may be used for building decision trees including Chi-squared Automatic Interaction Detection (CHAID), Classification and Regression Trees (CART), Quest, and C5.0 [1].

A main advantage of decision trees is that they can explain their classification process and the order in which input data are requested, and since they make few passes through the data, they are very efficient. They also handle categorical data very well. Decision trees are however susceptible to growth without bound and thus risk overfitting the data. They need to be controlled through *stopping rules* that limit their growth, for example by specifying a maximum depth or by establishing a lower limit on the number of records in a node. They are not allowed to split the tree below this limit. Another approach to limit a tree's growth is to prune the tree after allowing it to grow to a full size, using either heuristics or through user intervention.

Another disadvantage of decision trees is their inability to look ahead when choosing a split, and therefore may not consider different and sometimes better patterns for classification. Furthermore, most algorithms used in decision trees consider only one predictor variable at a time. This approach limits the number of possible splitting rules to test and makes relationships between predictor variables hard to detect.

7.4 K-Nearest Neighbor

K-nearest neighbor is a classification algorithm that classifies a new object by examining *k* of the most similar objects or neighbors. It counts the number of objects of each class, and assigns the new object to the same class to which most of its neighbors belong.

The application of the algorithm is as follows. First, the distance between the objects is calculated. Next, a set of already classified objects is selected as the basis for classifying new objects. The size of the neighborhood in which to do the comparisons is then selected, and the weight given to the different neighbors is decided upon (e.g., nearer neighbors are given more weight than farther neighbors). Figure 4 indicates a new object *N* would be assigned to the class *X* because the number of *X*'s within the neighborhood outnumber the number of *Y*'s [1].

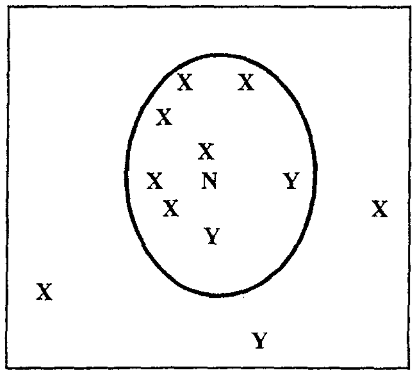


Figure 4. K-Nearest Neighbor. N is a New Object.

The challenge in applying the K-nearest neighbor algorithm is to find a suitable metric for measuring the distance between attributes in the data and then calculate it. Categorical variables present a particular challenge. For example, what is the categorical distance between a hostile submarine and a friendly merchant ship?

K-nearest neighbor is a computationally demanding algorithm as the computational time is proportional to the factorial of the total number of objects. Unlike a neural network or decision tree, the algorithm requires a new calculation to be made for each new object to be classified.

However, K-nearest neighbor models are very easy to understand when there are few predictor variables. They can also accommodate non-standard data types, such as text, as long as an appropriate metric is identified.

7.5 Genetic Algorithms

Genetic algorithms refer to a variety of problem-solving techniques that are based on models of natural adaptation and evolution [11]. They are designed the way populations adapt to and evolve in their environments. Members that adapt well are selected for mating and reproduction. The descendants of these members inherit genetic traits from both their parents. Members of this second generation that also adapt well are selected for mating and reproduction and the evolutionary cycle continues. After several generations, members of the resultant population will have adapted optimally or at least very well to the environment.

Genetic algorithms start with a fixed population of data structures that are candidate solutions to specific domain tasks. After requiring these structures to execute the specified tasks several times, the structures are rated for their effectiveness as domain solution. On the basis of these evaluations, a new generation of data structures is created using specific "genetic operators" such as reproduction, crossover, inversion and mutation. Poor performing structures are discarded. This process is repeated until the resultant population consists of only the highest performing structures.

Genetic algorithms are not used to discover patterns per se, but rather to guide the learning process in other data-mining algorithms such as neural nets. Whereas the novel approach of genetic algorithms is an interesting one, it requires significant computing overhead.

7.6 Other Models and Algorithms

Other models and algorithms have been suggested to mine data. Some represent new approaches, whereas others are simply variations of algorithms that have been published in computer science or statistics journals. These models and algorithms include Multivariate Adaptive Regression Splines (MARS), logistic regression, discriminant analysis, Bayesian networks, and Generalized Additive Models (GAM).

8. CONCLUSION

Knowledge discovery offers great promise in helping organizations to uncover hidden patterns in their data, to explain existing data and make predictions about new data. Although the selection of appropriate data-mining algorithms and models is a vital step in the success of the knowledge-discovery process, it is equally important to collect, cleanse, and integrate the data properly prior to the application of data-mining algorithms.

ACKNOWLEDGEMENTS

The authors thank the Office of Naval Research, the Space and Naval Warfare Systems Center, San Diego Science and Technology Initiative, and the Center for Reconnaissance Research, Naval Postgraduate School for financial support of this work. This chapter is the work of U.S. Government employees and no copyright subsists therein. It is approved for public release with an unlimited distribution.

REFERENCES

1. Introduction to Data Mining and Knowledge Discovery, Two Crows Corporation, 2000.
2. Batini C., Certi S., and Navathe S. B., Conceptual Database Design: An Entity-Relationship Approach, Benjamin/Cummings Publishing Co., 1992.
3. Ceruti M. G., Application of Knowledge-Base Technology for Problem Solving in Information-Systems Integration, Proceedings of the 14th DOD Database Colloquium '97, pp. 215-234, Sep. 1997.
4. Ceruti M. G., and Kamel M. N., Semantic Heterogeneity in Database and Data Dictionary Integration for Command and Control Systems, Proceedings of the DOD Database Colloquium '94, pp. 65–89, Aug. 1994.
5. Ceruti M. G., and Kamel M. N., Heuristics-Based Algorithm for Identifying and Resolving Semantic Heterogeneity in Command and Control Federated Database Systems, Proceedings of IEEE Knowledge and Data Engineering Exchange Workshop, KDEX'98. pp. 17–26, Nov. 1998.
6. Ceruti M. G., and Kamel M. N., Preprocessing and Integration of Data from Multiple Sources for Knowledge Discovery, in a special issue of International Journal on Artificial Intelligence Tools, (IJAIT), vol. 8, no. 2, pp. 152–177, June 1999.
7. Ceruti M. G., Rotter S. D., Timmerman K., and Ross J., Operations Support System (OSS) Integrated Database (IDB) Design and Development: Software Reuse Lessons Learned, Proceedings of the Ninth Annual AFCEA Database Colloquium '92, Aug. 1992.

8. Ceruti M. G., Thuraisingham B. M., and Kamel M. N., Restricting Search Domains to Refine Data Analysis in Semantic-Conflict Identification, Proceedings of the Seventeenth AFCEA Federal Database Colloquium and Exposition '00, pp. 211–218, Sep. 2000.
9. Fayyad U., Piatetsky-Shapiro G., Smyth P., and Uthurusamy R.., Advances in Knowledge Discovery and Data Mining, MIT Press, 1996.
10. Kamel M. N., Identifying and Resolving Semantic Conflicts in Distributed Heterogeneous Databases, Proceedings of the Tenth Annual DOD Database Colloquium '93, AFCEA, San Diego, CA, Aug. 1993.
11. Kamel M. N., Knowledge Acquisition, in Wiley Encyclopedia of Electrical and Electronics Engineering, J. G. Webster, Ed. John Wiley & Sons, vol. 11, pp. 107–122, 1999.
12. Mehrotra M., and Wild C., Multi-viewpoint Clustering Analysis, Proceedings of the 1993 Goddard Conference on Space Applications in Artificial Intelligence, pp. 217–231, May 1993.
13. Sheth A. P., and Larson J. A., Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, ACM Computing Surveys, vol. 22, no. 3, pp. 183–236, 1990.