Mining Mart: Combining Case-Based-Reasoning and Multistrategy Learning into A Framework for Reusing KDD-Applications

Jörg-Uwe Kietz[†] Regina Zücker[†] Anca Vaduva[‡]

[†] Swiss Life
IT Research & Development
CH-8022 Zurich, Switzerland
{Uwe.Kietz,Regina.Zuecker}@swisslife.ch

[‡] University of Zurich
Department of Information Technology
CH-8057 Zurich, Switzerland
vaduva@ifi.unizh.ch

Abstract

One of the most time consuming steps for the process of knowledge discovery in databases (KDD) consists in preparing the source data. On the one hand, problems are raised by the large amounts of heterogeneous data of doubtful quality to be processed. On the other hand, since no universal tool is nowadays available to deal with all the various analysis tasks required by a real application, many different types of data mining algorithms and tools have to be employed. Typically, they have very strict and specialized input requirements. In this paper, we propose a case-based-reasoning (CBR) framework for the pre-processing step of the KDD-process. The aim is less to entirely automatize pre-processing and mining tool selection, but mainly to support the reusing of work done for one KDD-task to similar ones. As a side effect, the integration of pre-processing operations supported by multiple machine learning (or data minig) will make the case adaption at least partially automatic.

1 Introduction

The entire (information) society is in a somewhat paradox situation: we are starving for knowledge while drowning in data. Traditional approaches fail to release the knowledge from the masses of available data. Two technologies are emerging to reduce the gap:

1. data warehousing and on-line analytical processing (OLAP) of data for the verification of hypotheses and
2. knowledge discovery in databases (KDD) for discovering new hypotheses.

Practical experience with these techniques have proven their value. However, it is also apparent that using a data warehouse for decision support or applying tools for knowledge discovery are difficult and time-consuming tasks. Therefore, it is currently still quite difficult to get a high return of investment from using them. Definitely, their main drawback is the pre-processing of data. If we inspect real-world applications of knowledge discovery, we realize that 50 - 80% of the efforts are spent on finding an appropriate transformation of the given data, finding appropriate sampling of the data, and specifying the proper target of data mining, etc. Furthermore, pre-processing is not only time-consuming, but also requires profound data mining and database know-how. As a result, pre-processing cannot be done by the common business people, but only by a few highly skilled power users.

A further problem is the mass of data to be pre-processed. Even if existing KDD-tools offer facilities to pre-process data so far, they fail in achieving this task for real applications because they cannot deal with large amounts of data. The reason is that data has to be first loaded from databases into the KDD-environment and then internally processed by making use of extensive temporary storage which becomes costly or even impossible for large amounts of data.

To overcome the shortcomings of the currently existing support for KDD, this paper addresses the following objectives:

1. Create a user-friendly KDD support environment (KDDSE) for the non-expert user. In particular, the focus lies on extending the already existing tools for data mining with a user-friendly environment for *data pre-processing*.

2. Provide reusable components for the expert-user within the pre-processing environment. These components may be either easily configured or effortlessly extended for implementing new data mining applications.

3. Avoid the limitations imposed by todays tools where the whole amount of data that has to be loaded, kept and internally handled within the pre-processing environment.

4. Speed up the discovery process by reducing the number and the complexity of trial and error pre-processing and analysis cycles.

In order to reach these objectives, we propose an application development framework called Mining Mart integrating techniques of CBR and multistrategy learning. The framework provides a collection of *cases* which may be either directly used and executed or reused for developing new ones. A case consists of the specification of a business problem, the data to be analyzed and a chain of pre-processing operators that are based on clever multistrategy learning. This framework may be used both, by experts and non-experts: the end-user retrieves one of the prepared cases, makes some simple adaption if required (e.g., the selection of another target segment) and initiates the case execution.

In contrast, the highly skilled data mining power-user uses the framework for creating new cases. The cases already available provide useful building blocks and analysis chains for reusing, which should speed-up the creation of new business cases. One of the particularities of our approach is the realization of the framework: unlike in other KDDSE, we let the data to be stored further on in databases and implement the pre-processing framework beyond the database management system(DBMS). In this way, we can make use of the inherent mechanisms provided by DBMS in handling large amounts of data. That means, the specification of the business problem and the pre-processing operators belonging to a case are (intensionally) stored in the form of metadata [Staudt *et al.*, 1999b; Staudt *et al.*, 1999a]. In particular, transformation specifications, database structures, configuration parameters and statistical information are stored with an appropriate granularity and in accordance with a suitable metadata schema in a repository. At runtime, the specifications are read, interpreted, merged into an operation chain and finally executed. The advantages are the *reusability* of metadata and the handling of *large amounts of data*: cases may be either directly used or easily edited and re-applied and - most importantly to tackle objective 3 - they may make use of the facilities offered by DBMS to execute the data transformations at runtime. In this case, the system is called *metadata-driven* and is based on a similar principle as the modern Extract/Transform/Load-tools like PowerMart (Informatica, http://www.informatica.com) or Datastage (Ardent, http://www.ardentsoftware.com).

	Time	Imp.
Business understanding	20%	80%
a) Exploring the problem	10%	15%
b) Exploring the solution	9%	14%
c) Implementation specification	1%	51%
Data preparation & mining	80%	20%
a) Data preparation	60%	15%
b) Data surveying	15%	3%
c) Modeling (data mining)	5%	2%

Table 1: Steps of a KDD-project with time to complete and importance to success

The remainder of the paper is structured as follows: Section 2 generally discusses aspects related to business cases, KDD-processes and their relation to data warehousing. In Section 3 we present our approach for data pre-processing. Section 4 presents related work while Section 5 concludes the paper.

2 Business Cases, KDD-projects and Data Warehousing

Ideally, a KDD-project starts with a business case, which could be solved or optimized by analyzing available data. The typical steps of such a project are given in table 1 from [Pyle, 1999]. First of all, it should be noted that the specification of *how to use* the expected mining results plays a decisive role for the success of a project and should be established in accordance with the company management at the very beginning; the best mining results are worth nothing, if they are not used. Furthermore, considering that the most time-consuming step is the preparation of data for mining (see table 1), both problems could be solved in a related manner. The management support for the use of the mining results as well as the justification of the high data preparation costs are both most easily reached if the KDD-project is integrated into a strategic and repeatedly occurring business case which may then reuse the pre-processing efforts each time it is executed.

For Swiss Life, a leading life insurance company of

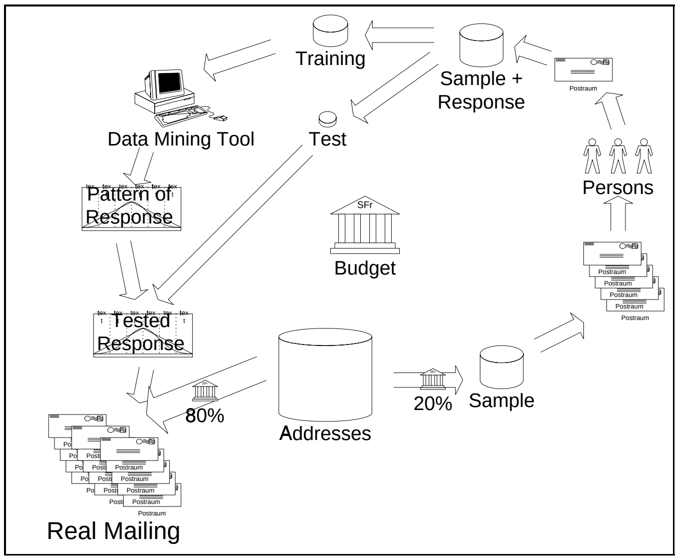

Figure 1: The Business Case Mailing Action

Switzerland, there are several application areas where central business cases could be supported by data mining [Staudt *et al.*, 1998], especially:

- Marketing
- Product development and controlling
- Business reporting

In this paper we will use the optimization of responses to mailing actions in direct marketing as an illustrating and well known example of such a problem. The aim is to perform mailing actions to those addresses with the highest probable rate of return. This business case[1] is illustrated in Figure 1. On a more technical level it may be described by the following steps:

0. Use an existing data warehouse (DWH) as base for extracting the needed data.
1. Construct a household view on this DWH, which provides all relevant information in a form, that allows the next step to be done by an end-user.
2. Select the target segment, e.g. households,
   (a) which have a child with an age below 2 and which are not mailed since the birthday of this child, or
   (b) which have already bought a single-premium insurance, but this is more than two years ago, or
   (c) ...
3. Select a random-sample from the target segment for training and test, let's say with size proportional to 20% of the budget. That means, if the budget is X and a single letter cost Y a total of X/Y letters may be sent. Spending 20% of the budget to get the training/test data means selecting X/Y*0.2 household records from the target segment.
4. Export the addresses of this sample, do the first mailing, and store the responses, i.e. label the sample.
5. Split the sample into training and test set.
6. Select/construct the relevant attributes for the current response prediction task.
7. Train the selected mining-tool, which output could be used to order (not just classify) the data.
8. Apply the data transformations done in step 6 to the test data (as the mined pattern relies on these data transformations)
9. Test the mined pattern on the test set, i.e get an estimated response rate for the mined pattern.
10. Apply the pre-processing done in step 6 to the target segment; the mined pattern relies on this preprocessing.
11. Select the best records (i.e., ordered by the mined reponse pattern) from the target segment of step 2. In accordance with the assumption in step 3, the selection set should be proportional to 80% of the budget.
12. Export the addresses of this selection and do the real mailing.
13. Compute a final evaluation, and store all the mailing-information (date, (non-) responses, segment, product, mined pattern, data-transformations, evaluation, ...) in the metadata repository, such that it could be used as background knowledge for further actions.

Considering the standard KDD-process [Brachman and Anand, 1996] in Figure 2 consisting of five phases, step 0 corresponds to the end of second phase. The building

[1]Ling and Li [1998] describe another problem and solution analysis of this business case. However their analysis is based on the assumption that existing customers have been acquired by mailings, (i.e., they all are persons answering to mailings), which is not the case in our setting, where most contracts are still sold by insurance-agents and not by mailing-actions.

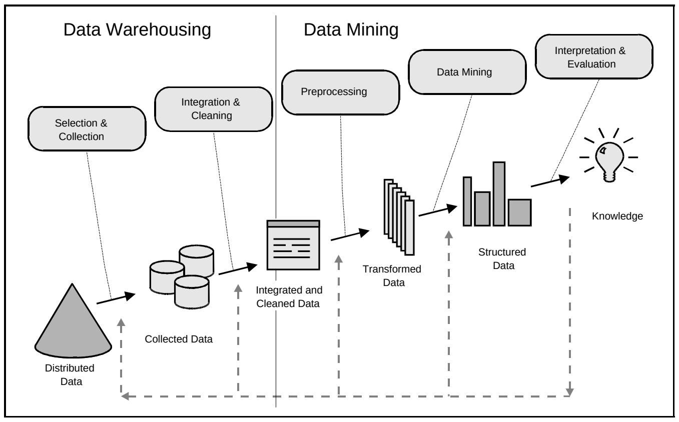

Figure 2: The KDD Process

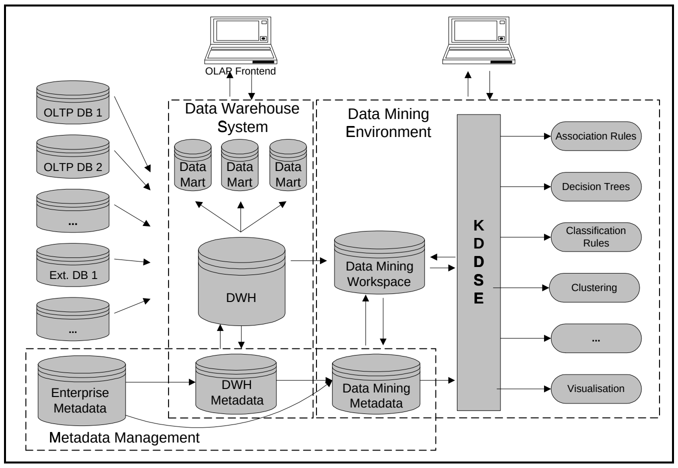

Figure 3: Architecture of a Data Warehousing and KDD Environment

of a data warehouse (DWH) covers the first two phases (selection & collection and integration & cleaning). Our approach relies on an already built DWH, which is, in fact, an ideal first step in setting up a KDD-process [Inmon, 1996]. The interplay between data warehousing and KDD is depicted in Figure 3. The data warehouse is build by extracting, transforming, integrating, cleaning, and finally loading the data from the operative systems. In the figure, these data sources are labeled as OLTP (on-line transaction processing) systems. While DWH is still a relational, normalized and a mostly general-purpose database, a data mart[2] contains application-specific and aggregated data for an OLAP(on-line analytical processing) application. Thus, data marts are not very useful for data mining, since they usually do not contain the data at the necessary level of detail needed for mining. For example, the typical data mart for the monthly product sales (number, value, ...) stored by region, product and time, could not be used to analyze customer behaviour with mining methods, as it does not contain any reference to specific customers, even if it is an aggregation of customers buying behaviour[3]. As a consequence, a data mining workspace has to be built on top of the data warehouse, independently of the other data marts. Note in Figure 3 that metadata of both data warehouse system and data mining environment have to be integrated into a (logically) central enterprise repository in order to manage the whole meta-information in a unified and consistent way. The repository stores the connections between the data warehouse and the data mining environment through corresponding links between metadata elements.

[2]Note that no agreement exists regarding the distinction between DWH and data mart in the DWH-literature.

[3]However, the hierarchical dimensions, e.g. for product, time and region, of the data mart could be very useful background knowledge for mining.

As far as relevant for this business case, the basic structure of our data warehouse is shown in Figure 4[4].

When we start the pre-processing phase on top of such a DWH, we are faced with

- a normalized, multi-relational database as data source, whereas most existing data mining algorithms are single-table based,
- many features and tables which are only partially relevant for the current business case,
- a data representation optimized for maintenance and not for optimal use within a specific data mining tool for a specific task (e.g., the DWH stores the birthdate but for data mining the age is required).

For our mailing-case, this means that the mapping between DWH data representation and the specification of a customer segment (step 2) is far from being trivial. So the first step (1) has to generate a more application-oriented view on top of the DWH schema. Inherently, it strongly depends on the business case to solve. Since we are interested in households in our business case, the corresponding view will reflect this. In contrast, the base-level for product development would not be households, but insurance contracts.

The construction of the view already specifies the first group of pre-processing steps to be handled (i.e., stored, retrieved and adapted) by our case-based-reasoning system. The other group is the one described in step 6. For

[4]Further information on this data warehouse, in particular a data extract, are available on our Web for further experiments. It can be obtained from http://research.swisslife.ch/kdd-sisyphus/.

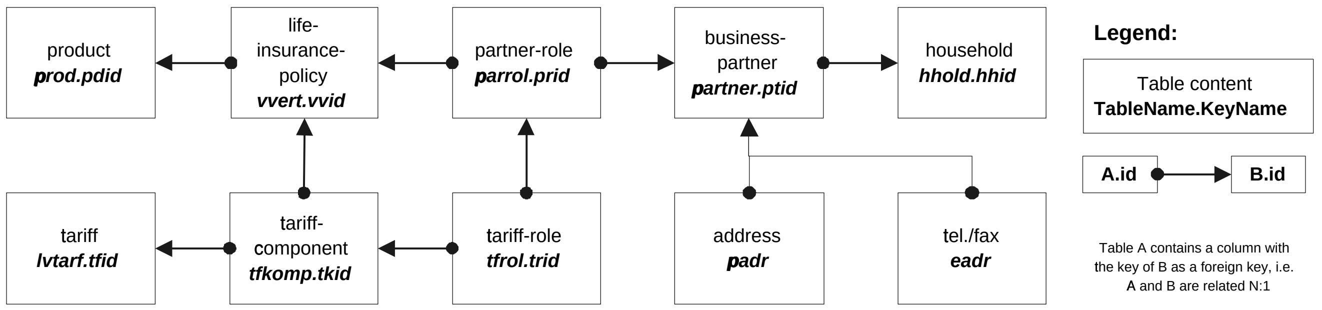

Figure 4: Schema excerpt from our DWH

the same business case, step 1 does not need much adaption for reuse since the same schema may be applied to another target segment (i.e. to 2.b instead of 2.a). However, this is not true for step 6 as different target segments may have very different properties, resulting in different attributes relevant for predicting behaviour: as a general rule it could be stated, e.g., that most households addressed by 2.a are not ensured so far, whereas for segment 2.b., the past insurance behaviour is very likely to be important for predicting future behaviour.

3 A Case-Based-Reasoning System for Pre-Processing

This section discusses our approach [5] for supporting preprocessing and implicitly data mining. As already mentioned, our aim is to build a system that supports the design, storage and management of business cases that may be directly used by end-users when solving a specific mining task. In addition, support for power-users is needed to describe and store new mining tasks.

Figure 5 illustrates the use of the CBR-system. An end-user comes up with a specific business task he wants to solve with data mining (e.g., doing the mailing action as described in the last section) and asks the system, if there exists a case for this task. If one of the cases fits user's intention, he can apply it, otherwise he has to contact a power-user to set up a new case.

To meet these requirements, a case must have two associated parts:

a) a business-case description for retrieval

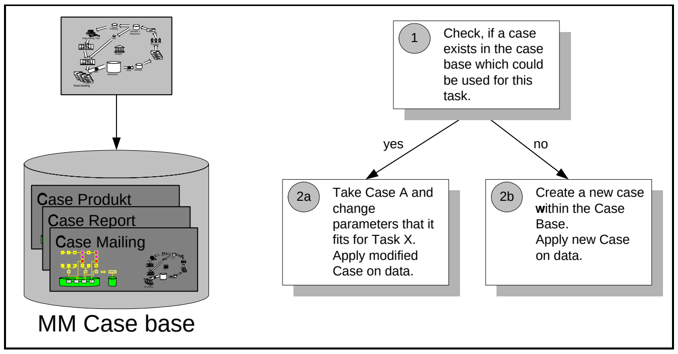
Case Produkt
Case Report
Case Mailing
MM Case base
1
Check, if a case exists in the case base which could be used for this task.
yes
no
2a
Take Case A and change parameters that it fits for Task X. Apply modified Case on data.
2b
Create a new case within the Case Base. Apply new Case on data.

Figure 5: The CBR Approach

b) a re-usable technical specification/implementation of the KDD-process for this business case (e.g., the machine-processable representation of the steps 1-12 of the mailing-case)

Since case-retrieval is not an issue of this paper, we assume that a case is manually picked from a given list[6], presented to the user by name or by a graphic like the one in Figure 1. In the following, we will limit our focus on the representation and re-use of the preprocessing operations for data transformations, i.e. the pre-processing steps 1, 2, 3, 5 and 6, and the related data transformation operations 8 and 10 of the mailing-case. We will neither cover the data mining (modeling) step itself (step 7 in the mailing-case) nor post-processing (testing/evaluation/use, steps 9 and 11) in this paper. However, at least clustering, classification learning and regression could be formalized as pre-processing operations constructing new attributes, i.e., the new class, the predicted class and the predicted value (see sec. 3.1.2) within our framework.

[5]The work is embedded in the Mining Mart project. Information may be found at http://www-ai.cs.uni-dortmund.de/FORSCHUNG/PROJEKTE/MININGMART/index.eng.html

[6]As long as we are limited to the mining business cases for the DWH of a single company or business unit this will probably be sufficient. For the whole Mining Mart project there will be a work-package concerned with business case description for retrieval.

- no 'unknown' (NULL) values are allowed for specific attributes
- scalar and ordinal attributes have to be numeric
- nominal attributes must have character values or be represented as sets of boolean values
- no numeric or no non-numeric attributes are admissible
- not more than N different values are allowed for nominal attributes
- always the same scale for numeric attributes is required
- no key attributes are respected
- input data must consist of a single flat table

Figure 6: Input restrictions of data mining tools

To start with, the goals of pre-processing are:

1. to provide the most relevant data for a certain task,
2. to provide the data in a form most suitable for mining,
3. to fulfill the input restrictions of data mining tools (Figure 6 lists some typical restrictions), and
4. to generate useful and necessary background knowledge to be used for future tasks

Inherently, pre-processing is embedded in a certain case. Important is, however, that the selection of the data mining tool is part of the case as well. This frees the end-user from the difficult task of selecting a data mining tool (see Section 4.2) and ensures at the same time that pre-processing exactly matches the input restrictions imposed by the tool (see Figure 6). That means, the tool is selected and integrated into a case when the case is designed. The use of the tool during case execution is transparent for the end-user.

In the following, we discuss how to adapt a case to reuse it successfully on the new data. As we already showed, even just the selection of another target segment from the same DWH possibly requires different relevant attributes to be considered. In our framework design, we integrate the approach of multistrategy learning (MSL) [Michalski, 1991; Michalski and Kaufman, 1998] to solve this problem of case-adaption to new data. In particular, we will combine constructive induction [Mehra *et al.*, 1989], with feature selection [Liu and Motoda, 1998b; Liu and Motoda, 1998a]. Additionally, several base feature construction operations are based on learning methods, e.g. discovering optimal discretizations and groupings.

At a first sight, MSL seems to be a very promising approach to total automation, as it does not need large amounts of currently unknown knowledge, but relies on systematic or heuristic exploration on the space of possible data transformations. However, pre-processing and mining real-world data using solely this approach is not practicable; even heuristic MSL approaches will possibly get lost in the huge search space of possible data transformations, and the advantage of not needing domain knowledge becomes the disadvantage of not being able to use such domain knowledge to restrict the search space to a manageable size. Our case-based approach to integrate pre-processing and data mining can be seen as an attempt to set up an environment where available knowledge can be used to specify some transformations and especially the structure of the case manually, and where multistrategy learning can be successfully used:

- to automatically solve manageable sub-problems where corresponding knowledge is not available, and
- to locally optimize the adaptation of the case to new data.

3.1 Atomic Operations of Pre-Processing

The base-operations of pre-processing investigated in this paper are sampling and segmentation, feature construction within a table and over related tables, and feature selection[7]. In our approach, feature construction and dropping of base-features are separated, as features may be needed for more than one feature construction operation (e.g. the date of birth of a person is needed to construct the age of the person, the entry-age into a contract, and the end-age of a contract).

3.1.1 Sampling & Segmentation

Sampling and segmentation are used to reduce the number of data records within the training data. The underlying goals of these operations are the following:

- Improve speed and reduce memory requirements of the mining tool
- Focus on rare or special cases
- Rebalance the class distribution
- Specify a target group
- Use only clean data

[7]In [Morik, 2000] the pre-processing of time data is investigated, which will be part of the future Mining Mart, too.

These operations leave the number of attributes and the data records unchanged, however the statistical properties of the population may change dramatically. Sampling can only be applied to one data table.

Examples

- Random sampling or splitting
  Extracts data records out of the set of training data by random generation.
  Parameters: number of records to extract.

- Sampling based on typical cases / atypical cases
  Extracts data records which fulfill the (a)typical case within the training data.
  Parameters: Condition of the (a)typical case.

- Segmentation
  Extracts data records which fulfill a special segmentation-pattern.
  Parameters: Segmentation condition (e.g. 2.a and 2.b).

- Data quality-motivated sampling
  Extracts only data records with values having a certain degree of quality.
  Parameters: Quality-condition (e.g. all records which have less than 2 unknown values).

- Rebalancing
  Noise-tolerant mining tools cannot be used to build a classification for very unbalanced class distributions (e.g. 95% – 5%, with 5% being an optimistic estimate for responses of a mailing.), a sampling favoring the members of the minority class has to be applied first.
  Parameters: class distribution (e.g. 55% – 45%) after rebalancing.

3.1.2 Attribute Construction within a Relation

Simple attribute construction creates a new attribute within one data table or view. The new attribute is based on one or more base attributes and groups their values into a more general form. The total number of data records is the same as before the operation. However, if base attributes are replaced with newly created ones, the number of distinguishable data records is possibly reduced (exception make relativation and rescaling).

Goals

- Improve data-coding relative to the capabilities of the mining tool
- Create a new attribute which can be better used for the mining task

Examples

- Discretization
  Is applied to attribute(s) of the type ordinal or scalar and creates a new attribute of the type nominal ordered
  (ordinal/scalar $\rightarrow$ nominal ordered).
  Parameters: base attribute(s), output attribute, number of created intervals for the output attribute (e.g. values of the income-attribute(s) are grouped into i0, i1..., i10).

- Grouping
  Is applied to attribute(s) of the type nominal which has/have no hierarchy and creates a new nominal unordered attribute.
  (nominal unordered $\rightarrow$ nominal unordered).
  Parameters: base attribute(s), output attribute, number of created groups, number of data records in one group (e.g. profession descriptions are grouped into groups of professions).

- Abstraction
  Is applied to attribute(s) of the type nominal or scalar and creates a new attribute of the type nominal ordered.(nominal, scalar $\rightarrow$ nominal ordered).
  Parameters: base attribute(s), output attribute, hierarchy, output of hierarchy level (e.g. looking at the household level instead of person level).

- Relativation
  Puts one attribute in relation to another attribute. This works only on numeric or date attributes and doesn't change the number of different data records.(numeric, date $\rightarrow$ numeric ordered).
  Parameters: base attributes, output attribute, operation between the base attributes (e.g. calculating the age from Sysdate and birthdate, calculating the quotient of income and premium sum).

- Cleaning
  Eliminates rare values of data records by creating a new attribute.(any type $\rightarrow$ any type).
  Parameters: base attributes, output attribute, which value of the base attribute shall be replaced by which new value (e.g. replacing the entry age of a person smaller than one by one).

- Unknown elimination
  Replaces unknown values with a specified new value.(any type $\rightarrow$ any type).

Parameters: base attributes, output attribute, specified replacing value (e.g. replacing the unknown values of attribute age by the mean value or most frequent value).

- Scaling
For all distance-based mining tools (e.g. clustering and instance based learning) the scale of the numeric attributes is very important, i.e. attributes with larger values are more influential on the result. To avoid this usually unintended weighting of attributes, all attributes have to be rescaled, e.g. to a fixed standard deviation or interval. (scalar$\rightarrow$ scalar).
Parameters: standard deviation or interval

3.1.3 Multi-Relational Attribute Construction

Joins and aggregations are used to put information from several related tables into one base table. To avoid unwanted changes of the target population distribution, the object identity within the base table has to remain unchanged (the number of different data records is still the same after the operations). To avoid the loss of base-table record-joins, outer-joins should generally be used, and to avoid the duplication of base-table records, simple joins must not be used for 1:N or N:M related tables (as this would duplicate records in the base table; e.g., it is not a good idea, to send duplicated mail to a household, for every person and contract involved). Instead the aggregation operations of this section have to be used[8].

<u>Goals</u>

- Fit the single table requirement of most DM-tools
- Reduce the complexity for ILP-DM-tools
- Avoid unwanted changes of the population distribution.

<u>Examples</u>

- Sum
Creates a scalar attribute, e.g. premium sum of a product, income of a household.
- Min, Max
Creates a scalar ordered attribute, e.g. smallest, highest premium sum of a product, smallest age of a member of the household.
- Count different
Creates a numeric attribute, e.g. how many persons a household has, how many insurance contracts a household has.
- Count X = V
Creates a numeric attribute, e.g. how many children has a household, how many different values has an attribute.
- Exist X = Y
Creates a binary attribute, e.g. exists an insurance contract of type 3a for one household.
- All X = Y
Creates a binary attribute, e.g. are all insurance contracts of a household of type 3a, are all persons of a household adults.

3.1.4 Attribute Selection

Attribute selection drops attributes which should not be in the mining input and/or result, since they are, e.g., clearly uninteresting, not usable or difficult or expensive to measure for new data. The number of different data records and also the number of the total data records remains the same.

<u>Goals</u>

- Drop attributes which do not fit the input requirements of a data mining tool
- Drop attributes which are strongly dependent on or the base of attribute construction for other attributes
- Improve processing speed
- Guide the build-in attribute selection process of the data mining tool

<u>Examples</u>

- Feature / attribute selection
Only the important attributes are chosen as input for the data mining tool. Selecting the attributes can be

[8]The design of these operations is in a way inspired by theoretical results in Inductive Logic Programming (ILP) on how to translate determinate hornclauses into propositional logic [Džeroski *et al.*, 1992; Kietz and Džeroski, 1994], which are also used in the ILP-system DINUS [Lavrač and Džeroski, 1994] for propositionalisation. Newer and more general propositionalisation approaches like [Alphonse and Rouveirol, 1999] are not used, as they are based on duplications of records in the base-table, which does not seem to be adequate in this context. Instead, we use operations inspired by Description Logics (DL) [Brachman and Schmolze, 1985] and the constructive induction operations of the DL-learning system KLUSTER [Kietz and Morik, 1994] to generate determinate features from indeterminate relations. These relations will also be further investigated in a future Mining Mart workpackage.

done manually, through input-restrictions associated with the mining tool or by feature selection methods [Liu and Motoda, 1998b; Liu and Motoda, 1998a].

3.2 Construction of Pre-Processing Cases

The development process of pre-processing cases is performed on training data. Each case has to be represented by a chain of several pre-processing operations having specific parameters and a defined order of their execution. An operation is either user-defined (so-called manual operation) or supported by an existing MSL tool. In order to develop an optimal pre-processing chain, the power-user has to execute several iterations to find the best fitting (global optimized) chain of pre-processing operations and MSL tools; the integrated MSL tools help him to automatically find a locally optimized solution. After all iterations are completed, several pre-processing chains with mining results exist. The chain with the best result has to be chosen as the best fitting one[9]. Then, this chain is executed on the test and application data as well (e.g., steps 8 and 10 of the mailing case).

A single iteration on the training data consists of the following steps:

1. Chain specification. MSL tools are selected and manual operations are either newly defined[10] or existing ones are reused. Operations have to be linked to build the chain: each takes as input the result of the previous operation and passes the output to the next operation.

2. SQL generation. Manual pre-processing operations are generated as SQL statements.

3. Chain execution. The execution of SQL statements is interwoven with calls of MSL tools which are standalone executables. The whole chain performs the required data transformations; at the end of execution, the data is ready to be loaded by the mining tool.

3.2.1 Handling Manual Operations

In Figure 7, the three steps are depicted for manual operations. During specification (1), the operation descriptions (PP) are read, decomposed and mapped into reusable

[9]For future versions the other ones could be stored as variants in the case base which help the adaption of a case to a new target segment.

[10]For supporting operator specification, a visual programming environment is planed in the Mining Mart project. At present, SQL is used.

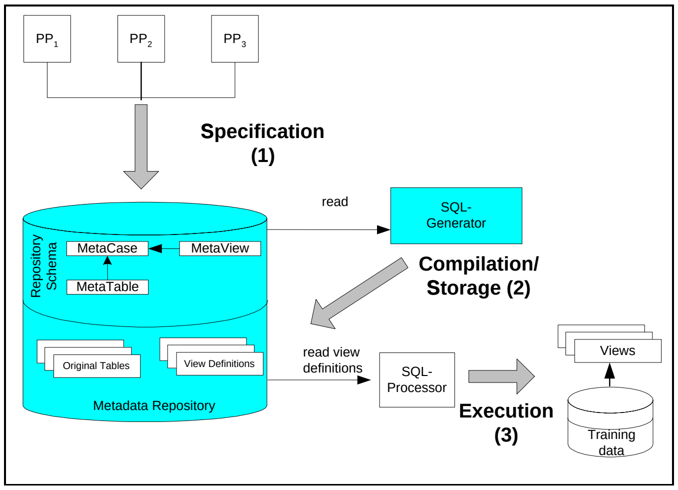

Figure 7: Development and execution of manual operations

SQL code parts (i.e., metadata elements) which are stored in a repository according to a suitable metadata schema. SQL statements are generated through the execution of a Java program which takes as input the metadata elements and brings them together in the required order, with the required syntax. For example, when creating a view, the metadata to be used includes the view name, view attributes, transformation functions applied to original attributes, constraints, the original tables with their description elements (name, attributes, primary and foreign keys), etc. The SQL generation, also called compilation (2), has as output an SQL statement which is stored into the repository as well. The execution (3) of the statement creates a view and either writes the definition of the view into the data dictionary of the training database or lets the view definition in the repository and only builds the (net) connection to the database to get the required data when the view is used (remote access).

Inherently, the repository also contains informations needed to run the MSL tools (paths, configuration files, etc). In this way, whole chains are stored into the repository; parts of them may be reused. Besides reusability, the use of the metadata repository for development and execution provides a better documentation. Operations, chains and tools may be accompanied by extensive descriptions of their tasks, executions, required parameters, etc. The fact that operations and their descriptions are managed together in an uniform way increases the software quality and implicitly its maintenance. Additional statistical information regarding the used data is also available in the repository, e.g., nominal and cardinal attributes, response-rate, etc. Furthermore, the repository stores descriptions of the business cases and the links to the op-

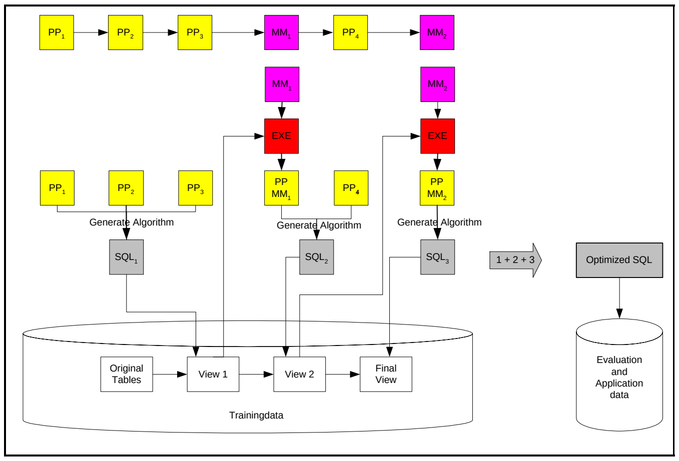

Figure 8: Chain of pre-processing operations

eration chains implementing them. This information is needed by the end-user to choose the right business case for execution.

3.2.2 Integrating MSL Tools into the Chain

Figure 8 shows a complete chain of pre-processing operations and MSL tools and also how SQL-statements are generated from that. The chain contains different manual pre-processing operations (PP) and MSL tool supported pre-processing operations (MM) in a specific order. The MSL tools are used to discover the parameters of the associated manual pre-processor operations (i.e., of their successors in the chain). Manual pre-processing operations are translated into SQL-code. MSL tools are stand-alone executables. The result of a SQL-statement is a view in the database which acts as an intermediate result, i.e. the view is needed as the input for a following MSL tool. In turn, the result of a MSL tool is used as input parameters of the associated SQL-code.

As an example consider the discovery of discretization parameters as the task of a MSL tool. For a given data set (i.e., the training data) and a number of discrete values (e.g., 2) the MSL tool finds the best mapping of intervals of the base-attribute (e.g., AGE) to nominal values of the new attribute (e.g., AGE_CATEGORY). The meaning of the mapping may be {if $AGE < 18$ then $AGE\_CATEGORY$ = 'CHILD', else $AGE\_CATEGORY$ = 'ADULT'}. That means, the next operation takes as input the old attribute, AGE, the automatically found parameter value, in our case 18, and the two discrete values, 'CHILD' and 'ADULT' (see Figure 9) and generates an (intermediate or final) view that has a new attribute AGE_CATEGORY taking only the

```
CREATE VIEW New_View (name,..., age_category) AS
SELECT name,..., Discretization(age,18,'ADULT','CHILD')
FROM Partner;

FUNCTION Discretization(age IN NUMBER, bound IN
NUMBER, value1 IN VARCHAR, value 2 IN VARCHAR2)
RETURN VARCHAR2 AS decoded VARCHAR2(50) :=NULL;
BEGIN
decoded:= DECODE(SIGN(age - bound), 1, value1,
-1, value2, value1);
RETURN (decoded);
END;
```

Figure 9: View with an attribute on which discretization has been applied

two discrete values.

After the whole chain is executed on the training data, the final view and several SQL-statements exist. Since the results of the MSL tools are now available, they may be embedded into the SQL statements. These may be merged and optimized to one statement (see Figure 8). This statement or the single generated SQL statements are executed on the test data (step 8 of the mailing case) or the application data (step 10 of the mailing case).

4 Related Work

Precondition for the success of the Mining Mart framework is the existence of a user-friendly and efficient KDD environment. We intend to achieve this precondition through a KDD support environment (KDDSE, [Brachman and Anand, 1996]) that advances the state of the art of current KDDSEs (Section 4.1). Such an advanced KDDSE will utilize research results for semi-automatic process planning support as discussed in Section 4.2.

4.1 KDD Support Environments

The KDD-process itself is nicely defined in the CRISP-DM process model [Reinartz *et al.*, 1998]. The pre-processing phase is the most time consuming task for practical applications. Therefore we discuss the support given by KDDSEs in this process. Data mining environments that support pre-processing generally only support internal pre-processing of data already loaded into the KDDSE. It is problematic, if not impossible, to force the content of a data warehouse into the KDDSE before applying the first pre-processing operator. However, with the announcement of the SPSS Clementine-Server and the better integration of DB2 and IBM's Intelligent Miner this situation is currently changing. Another problem is, that the result of a pre-processing operation is often stored extensionally. This makes the reuse of operations difficult and furthermore, if a problem compounded of a series of pre-processing operations is executed, several nearly identical copies of the original data set must be stored to prevent the loss of the intermediate steps required for further documentation and re-application purposes.

As an alternative, most KDDSEs allow the use of manually specified SQL expressions. However, specifying pre-processing operations in SQL is difficult, and the user is forced to select a sample of the database (via SQL) to be able to conduct the necessary pre-processing inside the KDDSE. The pre-processing environment to be developed in this project introduces support for in-database pre-processing operations. Particular research issues address the introduction of operations suited for multi-relational databases. The importance of support for multi-relational pre-processing is increasing with the introduction of commercial KDDSEs supporting multi-relational analysis (Kepler and Clementine after completion of the Aladin project). Multi-relational analysis is the next step in the evolution of KDDSEs towards allowing analysis of complex, large, and most importantly, relational company data warehouses.

4.2 Semi-automatic process planning support

Beginning with the MLT-Consultant [Sleeman *et al.*, 1989] there was the idea of having a knowledge based system supporting the selection of a machine learning method for an application. The MLT-Consultant succeeded in differentiating the nine MLT learning methods with respect to specific syntactic properties of the input and output languages of the methods. However, there was little success in describing and differentiating the methods on an application level that went beyond the well known classification of machine learning systems into classification learning, rule learning, clustering, and sloppy modeling. Also, the STATLOG ESPRIT-Project [Michie *et al.*, 1994], which systematically applied classification learning systems to various domains, did not succeed in establishing criteria for the selection of the best classification learning system. It was concluded that some systems have generally acceptable performance; and in order to select the best system for a certain purpose, they must each be applied to the task and the best be selected through a test method such as cross-validation. Theusinger and Lindner [1998] are in the process of re-applying this old idea of searching for statistical dataset characteristics necessary for the successful applications of DM-tools. An even more demanding approach was started by Engels [1997]. This approach not only attempted to support the selection of DM-tools, but built a knowledge-based process planning support for the entire KDD-process. Until today this work has not led to an usable system [Engels *et al.*, 1997]. The European project MetaL now aims at learning how to combine learning algorithms and datasets. We do not believe that this top-down knowledge-based approach will lead to an usable environment in the short run, as it requires a large amount of very application-specific knowledge. Furthermore, it is widely agreed upon that even the manual KDD-process cannot be planned ahead of time in detail. The utility of an operation can often only be determined after a large number of further operations is executed. It is apparent that not enough knowledge is available to propose the correct combination of pre-processing operations. However, it is possible to collect knowledge to exclude illegal, meaningless and unsuccessful combinations of operations. Therefore, we propose to use case-based semi-automatic process planning. Our goal is a system which supports a group of experienced data mining and domain experts in creating initial cases of the KDD-process for a specific application (e.g. the mailing action) and a specific type of data (e.g. a company's data warehouse). The system then offers these cases to domain experts, and supports them in repeating the KDD-process on new data of the same type (e.g. the same data warehouse, updated with new data, the result of the last mailing, etc.) for a similar application (the next mailing action). Hence, only the first use/creation of a case will require substantial effort and DM-experience, whereas all further uses of this case will be much quicker and more inexpensive. Additionally, a larger number of data mining applications can be executed with a limited number of available DM-experts. This case-base may even be useful to acquire knowledge about the KDD-process itself, e.g. by the Meta-Level Learning Methods developed in MetaL. This information could be utilized for more sophisticated process-planning support of initial cases for new applications.

5 Conclusion

The Mining Mart framework described in this paper builds on the insight that current approaches for achieving the objectives described above tend to ignore theoretical results, which have been proven that no algorithm can claim to be systematically better than any other on every problem [Wolpert and Macready, 1995], and that nobody has yet been able to identify reliable rules for predicting, that one algorithm should be superior to others, i.e. a total automation of the KDD-process is not possible.

A constraint based graphical user interface based on the KDDSE Kepler [Wrobel *et al.*, 1996] utilizing metadata shall guide users through the knowledge discovery task. The highest possible degree of automation for this process will be the aim of this project. However, as reasoned above, it cannot be expected that the user simply asks a high level question and selects a data set to be analyzed and everything else is done automatically. In particular, the task of proper transformation of the given data into a format that can be successfully analyzed by the available algorithms is difficult. As discussed above, testing of all possible approaches through pure multistrategy learning is currently not practical because the required computational power is not accessible for any single user. However, user that have access to the Mining Mart can search the case-base for suitable solutions to their task at hand. If no proper solution is found, the task will be posted as a new challenge to the knowledge discovery experts.

The main innovation of this project will be the deep integration of the different research directions currently accessible only to experts into an uniform environment usable also by data mining non-experts.

Acknowledgements: This work has been partially found by the Swiss Government under the contract-no "BBW Nr.99.0158" as part of the European commision Research Project IST-1999-11993 (Mining Mart). We thank Cèline Rouveirol and Ulrich Reimer for very helpful comments on a draft of this paper and all partners of the mining mart project with whom we had fruitful discussions about the framework described in this paper.

References

[Alphonse and Rouveirol, 1999] E. Alphonse and C. Rouveirol. Selective propositionalization for relational learning. In *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*. Springer Verlag, 1999.

[Brachman and Anand, 1996] R. Brachman and T. Anand. The process of knowledge discovery in database. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171 – 216, 1985.

[Džeroski *et al.*, 1992] S. Džeroski, S. H. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proc. Fifth ACM Workshop on Computational Learning theory*, pages 128–135. ACM Press, New York, 1992.

[Engels, 1997] R. Engels. Planning tasks for knowledge discovery in databases; performing task-oriented user-guidance. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 170. AAAI Press, 1997.

[Engels *et al.*, 1997] R. Engels, G. Lindner, and R. Studer. A guided tour through the data mining jungle. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 163. AAAI Press, 1997.

[Inmon, 1996] W. H. Inmon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49–50, November 1996.

[Kietz and Džeroski, 1994] J.-U. Kietz and S. Džeroski. Inductive logic programming and learnability. *SIGART Bulletin*, 5(1), 1994.

[Kietz and Morik, 1994] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–217, 1994.

[Lavrač and Džeroski, 1994] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, Chichester, England, 1994.

[Ling and Li, 1998] C. X. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 73–79. AAAI Press, 1998.

[Liu and Motoda, 1998a] H. Liu and H. Motoda. *Feature Extraction Construction and Selection, A Data Mining Perspective*. Kluwer Academic Publishers, 1998.

[Liu and Motoda, 1998b] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[Mehra *et al.*, 1989] P. Mehra, L. Rendell, and B. Wah. Principled constructive induction on decision trees. In *Proc. of the Eleventh Int. Joint Conf. on AI (IJCAI-89)*, pages 651–656. Morgan Kaufman, Los Altos, California, 1989.

[Michalski, 1991] R. S. Michalski. Inferential learning theory as a basis for multistrategy task-adaptive learning. In *Proceedings of the first International Workshop on Multistrategy Learning*, pages 3 – 18. George Mason University, 1991.

[Michalski and Kaufman, 1998] R. Michalski and K. Kaufman. Data mining and knowledge discovery: A review of issues and a multistrategy approach. In R. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining Meathods and Applications*. John Wiley & Sons LTD, Chichester, England, 1998.

[Michie *et al.*, 1994] D. Michie, D. Spiegelhalter, and C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Chichester, England, 1994.

[Morik, 2000] K. Morik. The representation race – preprocessing for handling time phenomena. In *Proc. of the European Conference on Machine Learning, ECML-2000*. LNAI, Springer Verlag, 2000.

[Pyle, 1999] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, San Francisco, California, 1999.

[Reinartz *et al.*, 1998] T. Reinartz, R. Wirth, R. Clinton, T. Khabaza, J. Hejlesen, and P. Chapman. The current crisp-dm process model for data mining. In F. Wysotzki, P. Geibel, and K. Sch"adler, editors, *Beitr"age zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98)*. Technical Report 98/11, Technical University Berlin, 1998.

[Sleeman *et al.*, 1989] D. Sleeman, R. Oehlman, and R. Davidge. Specification of consultant-0 and a comparision of several learning algorithms. Mlt-deliverable d5.1, Machine Learning Toolbox Esprit Project P2154, 1989.

[Staudt *et al.*, 1998] M. Staudt, J.-U. Kietz, and U. Reimer. A data mining support environment and its application on insurance data. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 105–111. AAAI Press, 1998.

[Staudt *et al.*, 1999a] M. Staudt, A. Vaduva, and T. Vetterli. Metadata management and data warehousing. Technical report, Information Systems Research, SwissLife, 1999. http://research.swisslife.ch/Papers/papers.htm.

[Staudt *et al.*, 1999b] M. Staudt, A. Vaduva, and T. Vetterli. The role of metadata for data warehousing. Technical report, Information Systems Research, SwissLife, 1999. http://research.swisslife.ch/Papers/papers.htm.

[Theusinger and Lindner, 1998] C. Theusinger and G. Lindner. Benutzerunterstützung eines kdd-prozesses anhand von datencharackteristiken. In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98)*. Technical Report 98/11, Technical University Berlin, 1998.

[Wolpert and Macready, 1995] D. Wolpert and W. Macready. No free lunch theorems for search. Techinal Report SFI-TR-95-02-010, Santa F Institute, Santa F, CA., 1995.

[Wrobel *et al.*, 1996] S. Wrobel, D. Wettschereck, E. Sommer, and W. Emde. Extensibility in data mining systems. In E. Simoudis and J. Han, editors, *Proc. of the 2nd Int. Conf. On Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, USA, 1996.