How to preprocess large databases*

Regina Zücker, Jörg-Uwe Kietz

Swiss Life, IT Research & Development (CC/ITRD),
CH-8022 Zurich, Switzerland
{Regina.Zuecker, Uwe.Kietz}@swisslife.ch

Abstract. One of the most time consuming steps for knowledge discovery in databases (KDD) consists in preparing the source data. In real-world applications we have to deal with very heterogeneous data of doubtful quality as well as with a huge amount of data records. On the one hand there are various types of data mining algorithms available to solve a specific mining task, with typically very strict and specialized input requirements. On the other hand a database exists with some million data records and some thousand attributes. A difficult task, which mostly has to be performed manually, is to select the right attributes for mining. The attributes have to be relevant for the result of the mining task and must be qualified for mining, e.g. less than 50% null-values. This problem description addresses the preprocessing phase of a KDD process for a mailing action and points out some problems we have encountered.

1 Introduction

At Swiss Life, we are working on the mining task for improving the response rate of mailing actions[2]. We have a data warehouse with data about a person, e.g. age and sex of the person, kind of single life insurance, insurance sum, premiums, etc. Also, within this data warehouse data of all persons living in Switzerland are stored. These data contains information about a household, e.g. how many members a household has, information about where the household is located, etc. On the other hand there exists data about a past mailing action and their responses. The business task is to increase the response rate for further mailing actions by using the experience of the past mailing. The technical task is to store the necessary preprocessing and mining steps in such a way, that this information can be adapted and reused easily[4].

Preprocessing of millions of data records is not only time consuming but also very challenging. Following questions have to be solved:

1. Which attributes can be used?
2. How can they be transformed in a most useful format?

* This work has been partially funded by the Swiss Government under contract "BBW Nr.99.0158" as part of the European Commission Research Project IST-1999-11993 (Mining Mart).

3. Which attributes have to be constructed and how?
4. How can this be done in a most adequate manner due to time and work expenses?

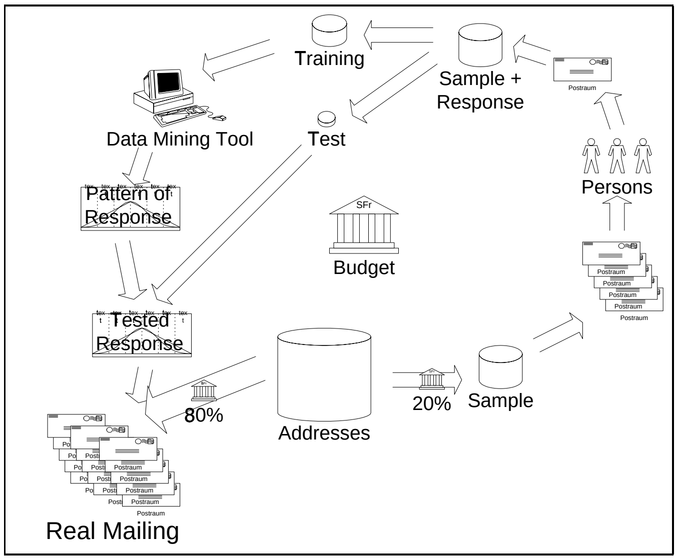

Fig. 1. The Business Task Mailing Action

2 The Business Task

Ideally, a KDD-process starts with a business task. For a mailing task in general the business task, which is illustrated in Figure 1, is defined by the following steps:

0. Use an existing data warehouse (DWH) as base.
1. Construct a household view on this DWH, which provides all relevant information in a form, that allows the next step to be done.
2. Select the target segment, e.g. households,
   (a) which have a child with an age below 2 and which haven't been mailed since the birth 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, with size proportional[1] to 20% of the budget, i.e. generate a sample to gather labels for the training and test set.
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 (preprocessing) done in step 6 to the test-data as the mined pattern relies on it.
9. Test the mined pattern on the test-set, i.e get an estimated response rate for the mined pattern on the target-group.
10. Apply the data-transformations (preprocessing) done in step 6 to the target-segment as the mined pattern relies on it.
11. Select the best (ordered by the mined response-pattern) records (proportional to 80% of the budget) from the target segment of step 1.
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 DWH/DWH-meta-data-repository, such that it could be used as background knowledge for further actions.

The business task at Swiss Life is based on a past mailing action. Therefore not all steps, which are described before, have to be applied. Step 3 and 4 are not necessary, because the data of the past mailing is equivalent to the sample of a first mailing. Steps 11 and 12 can also be skipped because the real mailing is already done. But at the end with step 13 all preprocessing steps of a mailing action are stored in a meta-data-repository and can be used and adapted for further mailings.

So far we have developed a household view (step 1), which contains 163 attributes and about 300.000 records. The original data has about 3.000.000 household records and about 5.500.000 person records. The view itself and the process to develop the view is stored as meta data in a database.

3 Preprocessing within a large database

As pointed out before, preprocessing is a challenging task, especially in large databases. It is not only time consuming to specify a preprocessing operation but also to apply it on a large database. Reusability of successful preprocessing steps for further mining tasks becomes more important[3]. Considering the whole KDD process, following requirements are essential:

[1] If your budget is X and a single letter costs Y you can send all in all X/Y letters, spend 20% of the budget to get the training/test data, i.e. select X/Y*0.2 household-records from the target segment.

1. Documentation of the KDD process
2. Reapplication of the KDD process to
   (a) evaluation data
   (b) application data
   (c) new data
3. Adaption of a KDD process to new business tasks
4. Process applicability to unlimited data sizes (execution within a database server)

Following we investigate the preprocessing tasks, which were necessary to fulfill step 1 of the mailing case, with regard to develop a meta-data repository for all necessary preprocessing operations not only the used ones of step 1.

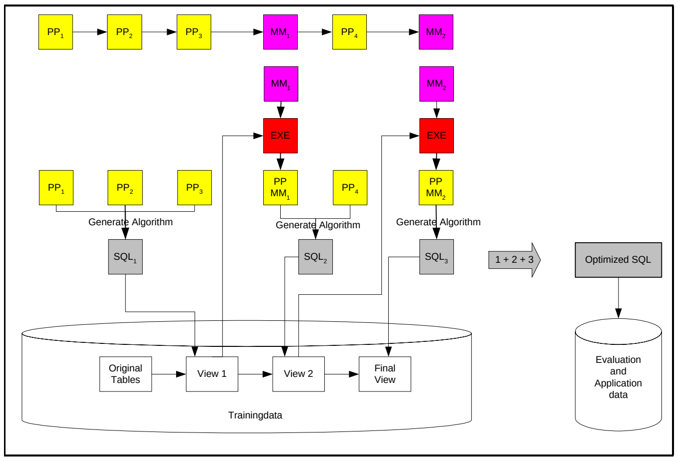

Fig. 2. Chain of preprocessing operations

All preprocessing operations of the mailing case can be considered as a chain with a special order. This is illustrated in Figure 2. Every single preprocessing operation has two parts, which are both stored within the database :

1. Specification of the operation within the meta-data repository.
2. The compilation result produced from the specification (either an operative view or a set of procedures, which apply the operation on many data records).

Figure 3 shows the two parts of a preprocessing operation as well as the requirement of minimizing the data within the data mining environment by transferring data into the database. In this approach the database is not only used as storage for original data which has to be mined but also as storage for the mining process (with all preprocessing steps) itself. Especially for handling a huge amount of data records in an acceptable time period it is necessary to

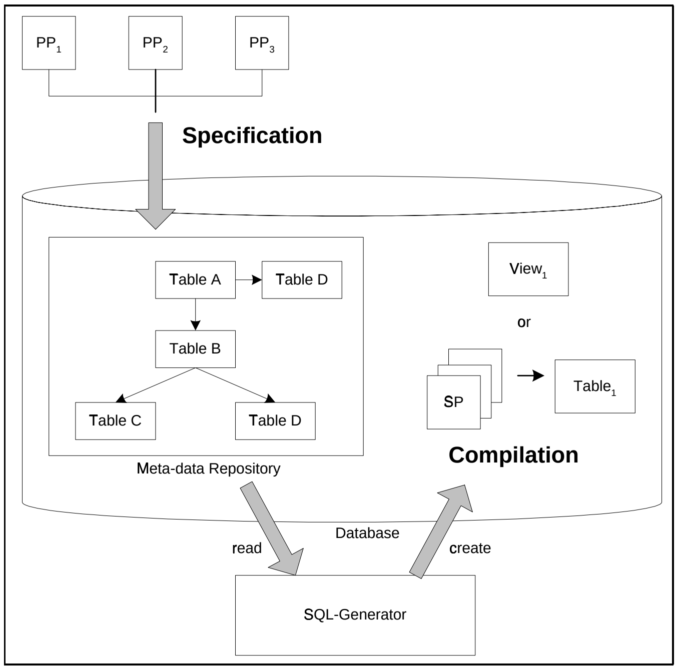

Fig. 3. Two parts of a preprocessing operation

apply algorithms within the database.

So far we have developed a database schema to store data about the data, e.g. a meta data schema, where every preprocessing step is stored. The implementation is done by creating a view over the original data from the data warehouse. If an attribute needs a transformation a database function (build in from the DBMS or developed by your own) is applied. The function is used within the view definition. With this method it is possible to transform an attribute several times by applying more than one view on it. We have developed a generator which reads the meta data information and creates the view within the database. As a first step on the way to have practical meta data, i.e. meta data about when which operation should be applied, statistic information about the original data is calculated and stored in the meta data schema. For that we have developed a procedure, which is also stored in the database.

4 Identified Problems

1. *Relevant attributes*
   In a first approach every attribute of the relevant original database table is registered in the meta-data repository. Then the statistics are calculated as a decision base for selecting the attributes with good values, e.g. no null

values or value distribution.

2. *Correlations/dependencies between attributes*
   Are there correlations or dependencies between attributes and how can they be detected? Up to now we only considered univariate statistics for attribute selection.

3. *Many data records in the original database*
   It's very time consuming to get the result of a view selection when the underlying table has some million records, especially when it is a remote table. If the view definition includes a join over more huge tables or the view is based on another view and then on a huge original table, the database is not able to create a result at all. In the first case the preprocessing step has to be splitted into several views instead of one. In the second case intermediate tables have to be created instead of a view. There is no automation available for the decision of creating a view or a table[2].

4. *Creation of new attributes*
   When a new attribute is created by transforming one or more attributes into a new one, the transformation process has to be stored so that it can be applied on test and application data as well. For attributes, which are based on a single attribute, this is solved by defining a database function and using it in the view specification. How can this problem be solved for attributes, which are based on more attributes?

5. *Reuse of preprocessing efforts*
   How can an existing preprocessing chain be adapted to a new business task? This has to be done in aspect of investing as less time as possible to get a good result. In [2] a case-based reasoning system is introduced as a first approach to reuse preprocessing operations. But until now no solutions can be presented for chain-adaption. What kinds of adaption exist and which can be applied to an existing chain? What are the arguments for developing a new preprocessing chain rather than adapting an existing one? For these questions no approaches exist yet.

6. *Relational database*[1]

   (a) The optimizers of relational databases are tuned to single row fetches, therefore selecting huge cursors of data records can end with an execution error, even when the statement is correct. Mostly the only possible

[2] To select count(*) of our developed household view, which is based on several views and intermediate tables, but no remote tables, it takes about 30 minutes to get a result, to select the first 20 rows it takes about 60 minutes. These measurements are sincerely dependent on the performance and utilization of the database server, but give an impression of the time scope.

solution is to create a table for storing intermediate results instead of a view. But then the recency of the table content has to be ensured.

(b) Inserting or deleting many data records will cause execution errors when not ensuring commits within a transaction. Therefore stored procedures have to be created for administering tables.

(c) Constructing a new attribute while aggregating an original attribute with more than one result record is only possible via original grouping functions of the database, e.g. min, max, sum, etc. According to the present experience it's not possible to self define a grouping function. Therefore important aggregation information cannot be created.

7. *Changes in the original database schema*
   Right now the contents of the meta-data repository are based on the schema of the data warehouse. In case the original schema will change, the adaption of the whole meta-data repository is necessary. So far this must be done manually. For the future the introduction of a mapping layer between the original data schema and the meta-data repository is conceivable. This layer can guarantee the independence of any original database schema.

5 Open Questions

As described before to apply only step 1 in our mailing case many problems emerged. For most of them we implemented a solution, which is applicable on huge data sets. But many execution steps are not based on a result of a meta learning method but on a pragmatic approach. Following open questions arise:

1. How to advice preprocessing step selection, e.g. by meta-learning?
2. What knowledge about a preprocessing operation and their context of application is necessary to support selection advice? Can this knowledge about the data to preprocess be acquired by learning?
3. How can the reuse of preprocessing effort be supported, if the context changes, e.g. the tackled business case or the database schema?

References

1. K. Loney G. Koch. *ORACLE8: The Complete Reference.* Osborne McGraw-Hill, 1997.
2. R. Zuecker J.-U. Kietz. Mining Mart: Combining case-based-reasoning and multi-strategy learning into a framework to reuse kdd-application. In *Proc. of the Int. Conf. on Multi-Strategy Learning, MSL-2000*, 2000.
3. J.-U. Kietz, R. Zuecker, A. Fiammengo, and G. Beccari. Enabling end-user datawarehouse mining. In *Data Sets, Meta-data and Preprocessing Operators at Swiss Life and CSELT, Deliverable 6.2*, June 2000.
4. K. Morik. The representation race - preprocessing for handling time phenomena. In *Proc. of the European Conference on Machine Learning, ECML-2000.* Springer Verlag, 2000.