Using Ontologies in a KDD Workbench

Timm Euler and Martin Scholz

University of Dortmund
Computer Science VIII
D-44221 Dortmund, Germany
{euler, scholz}@ls8.cs.uni-dortmund.de

Abstract. Domain understanding is one of the first steps in real-world applications of Knowledge Discovery in Databases (KDD). Preprocessing the data properly and finding an appropriate representation is crucial for a successful application of data mining algorithms. We present a metamodel of KDD preprocessing chains that contains an ontology for describing conceptual domain knowledge. This metamodel is operational, yet abstract enough to allow the reuse of successful KDD applications in similar domains.

1 Introduction

This paper presents the use of ontologies in the KDD preprocessing tool MiningMart[1] [4]. We describe the different goals that MiningMart addresses and how they are supported by the ontology formalism it is based on.

Data made available for KDD applications has often primarily been collected for other purposes. As a consequence data mining tools are hardly applicable to the raw data directly. One of the first steps foreseen by the cross-industrial standard for data mining (CRISP, [1]) is domain understanding. In this phase data analysts try to identify and understand the important concepts and relationships of the domain and to couple this information with the available data. Subsequent steps of the CRISP model augment the domain model, so that a representation of the data is achieved that can be fed into a mining algorithm.

The position taken in this paper is that establishing a higher level of data abstraction allows to ease the KDD process in terms of understandability and reusability, if working at this level is supported by a suitable tool with a graphical user interface. Well-known formalisms for data abstraction like entity relationship diagrams or UML's class diagrams allow to model concepts and some of their properties intuitively. However, the main purpose of such formalisms is to ease the construction of schemas by means of visualisation. Once the schema has been established it can be considered as an interface for inserting, updating and deleting objects, thus for applications the ideal case is a schema that does not change while it is in use.

[1] http://mmart.cs.uni-dortmund.de

In contrast to this, the CRISP model suggests an iterative nature of the KDD process, in which it is common practice to repeat experiments in a trial-and-error-like fashion. Operations like feature construction, feature selection, and aggregations of data sets can be considered to change the part of the database schema which is subject to the analysis. However, the original data and its schema do not have to be changed in any way. Thus we may look for alternative ways of abstraction for building data models in KDD.

Similar to the well known TCP/IP model, where different levels of abstraction are available for different kinds of services, we propose an abstract *conceptual* level on top of the logical level of relational databases, tailored to ease the development and re-use of KDD processes. Basically this level should enable experts to comfortably describe sequences of pre-processing and data mining steps, using understandable notions of the respective domain language. When analysing customer data, for example, a domain expert may probably prefer to think of customers as basic entities, although inspecting databases of different companies may reveal that names, adresses, contract information, or data about past interactions with customers are hardly ever stored in the same single table. Yet, at a higher level, it is possible and preferrable to describe such data in terms of the entities that are most natural to the domain expert. In this example, providing the concept "customer" as an entity on which further processing is based would help the expert to structure the complete application intuitively. This argument is similar to those for ontology-based query formulation in systems like OBSERVER [3]. The following sections explain the MiningMart approach in this light.

2 MiningMart Overview

A complete KDD process begins on raw data from an institution's databases, transforms the data into a format suitable for data mining (*preprocessing*), applies a mining algorithm and deploys the mining results on new data. The success of the mining algorithm depends strongly on the representation of the examples it learns from; this representation is built during preprocessing. It is a difficult task to choose a suitable mining algorithm and example representation, given only the initial data.

The MiningMart environment [4] is a graphical tool based on relational databases. It addresses the difficulty above by enabling the exchange of knowledge about successful KDD applications. To this end, such processes and their data are stored in a meta model called M4 [2]. By making M4 operational, the MiningMart system supports not only the storage, documentation and exchange of such processes but also their initial development and execution.

M4 consists of a data model and a case model; both of these parts are explained in the following.

1. The data is modelled at two levels. First, the database schema, which describes the tables, their attributes and links between tables, is stored. This allows a consistent access to this information across databases. Second, an

ontology level is introduced that allows to describe the data in more abstract terms. Basically, the ontology level uses *concepts* with *features*, and *relationships*, to model the data. Both levels are described in more detail in section 3. Obviously, the mapping between the two levels is crucial and section 3 will say more about this as well. The principal advantage of this two-level data model is that all the data processing will be described in terms of the ontology level, which allows to re-use the complete description on a new database by simply changing the initial mapping.

2. A complete sequence of operations describing a KDD process is called a *case* in MiningMart. The *case model* describes the operations executed on the data by providing an open set of fixed *operators* that perform basic data transformations as well as more sophisticated learning steps (sometimes learning is applied during preprocessing). Apart from some optional operator-specific parameters (constants) all inputs and outputs of operators are specified in terms of the domain ontology. In valid operator sequences outputs of predecessors are available as input to subsequent steps. As soon as the inputs of an operator have been mapped to database objects it becomes executable. Each operator has a specific task which is basically to establish a new database view on the given input data. This process can be seen as a transformation into a different data representation. The M4 *compiler* is the system component responsible for executing operators. It reads the information on how to apply operators in the current case from M4 and creates views based on dynamically generated SQL code.

The MiningMart environment includes a GUI that allows to a) create and edit objects at the ontology level and map them to database objects, and b) create and edit chains of operators. Further, all M4 information related to one case can be imported and exported to XML files; these files can be exchanged between users, either directly or using the central MiningMart web repository of cases.

To sum up, MiningMart is a KDD tool that makes use of ontologies in its data model. The main advantages of this approach are:

- Description of the data in terms familiar to the user
- Automatic documentation of data and processing steps
- Re-usability of KDD applications on different databases
- Exchangability of knowledge about successful KDD applications

The following section will provide more details about the data model in M4 and discuss several issues related to it.

3 MiningMart Data Model

3.1 An example

In this subsection the M4 data model is illustrated along a short example, taken from a telecommunications company that has modelled a customer relationship

Table 1. A Call Details Table.

CallerNumber	CalledNumber	Length	Date	Tariff	...
7222277	2777722	194	12-02-2002:18:04:56	11	...
1881181	8118818	82	24-12-2002:11:44:23	2	...
...	...	...	...	...	...

management application with MiningMart. Their database stores each telephone call individually; see table 1. For each call, the column *CallerNumber* contains the caller's telephone number, *CalledNumber* is the telephone number that was called, *Length* is the number of seconds the call took, *Date* gives the exact date and time of the call and *Tariff* gives a code for the tariff used for billing the call.

The lower level of the M4 data model, which represents the database schema, stores the table name (say `CallDetails`) and the names and datatypes of its columns (`CallerNumber`, `Length`, etc.). In addition, primary and foreign keys and information about cross tables can be stored.

At the ontological level, we might introduce a concept `Phonecall`. Each concept has got a non-empty set of *features*; for `Phonecall`, some features could be `CallingPerson`, `CalledPerson`, `Duration` and `Date`. `Phonecall` might be linked to another concept `Customer` by a relationship `Calls`. Inheritance of concepts is supported by the M4 formalism: the concept `Customer` could have subconcepts such as `Private Customer` and `Business Customer`. Subconcepts inherit all features of their super concept but have a smaller extension.

3.2 Mapping the logical to the conceptual level

In the above example, we have a simple 1:1 mapping between a concept and a database object (a table) that contains the data for that concept. Yet this mapping can simplify things for the user if not all attributes of the table are relevant for an application. In this case it is possible to introduce a concept that hides the irrelevant subset of attributes. But let us discuss some cases of more complex mappings.

We may want to have several concepts that use the same database table (using the same or different sets of columns as features). These concepts can serve different purposes in the KDD application in question. This is easily possible in MiningMart.

We may want to have concepts that use only parts of the contents of a table. For this, we may or may not want to introduce a concept for the complete table, and model subsets as subconcepts. For example it may be interesting to distinguish between several different kinds of customers. In any case, the subset of the table to be used must be specified. Since this kind of specification is provided as a data processing operator in MiningMart, it seems easiest to use the MiningMart environment to produce the concept desired; this can be done in very few processing steps. In fact, the MiningMart system will create the desired concept as a subconcept of the original concept whenever selection operations

are applied. As a special case, MiningMart offers parallel execution of processing operations on all subsets of a table if the subsets are distinguished by the values of just one attribute (Segmentation).

We may want to subsume two or more tables under the same concept, using all or a subset of their attributes. On the database level this corresponds to a table join. Joining tables to one concept makes most sense if there is a 1:1 relationship between the tables, that is, each row in one table corresponds to a single row in the other table(s). However, even with 1:n or n:m relationships this may make sense, especially when combined with further selection operations. The point is that in MiningMart, the necessary operations are provided anyway as part of the case model, that is, as data processing operations which may be needed in any KDD application.

To sum up, because M4 is used in an application that offers data manipulation operations such as joining and selection, it was easy to introduce a two level approach in the M4 data model, with the advantages listed at the end of section 2. The mapping between the two levels can be kept rather simple. Even so, rather than creating a redundant description layer for the *existing* database, as is necessary in ontology-based systems for interoperability of databases like [3], the extra level is used in MiningMart to describe a view of the data that is *wanted*, either to enable further processing in the KDD application at hand, or to model more naturally a human's understanding of the domain. The wanted view of the data can be created using the operations the system provides. This is especially interesting when a case model is ported to a new database. Semi-automatic schema matching approaches as described in [5] could simplify the adaptation of successful cases to new databases in the future.

3.3 Storing the meta model

As explained above, the M4 meta model that underlies the MiningMart system stores information about preprocessing steps and data at two levels of abstraction. All M4 information is stored in tables of a relational database. This choice was made because a database interface was needed anyway, and the additional powers of other formalisms like description logics are not needed for MiningMart's purposes. It also offers some advantages compared to, for example, flat-file storage using XML. Most importantly, it allows to build the meta model on top of the well understood relational calculus, with clearer semantics than most XML-based formalisms. The DBMS may be seen as a virtual SQL machine. It takes care about data consistency, transaction management, and optimisation of assigned resources. Especially the consistency of the meta data is of high value, because M4 contains highly structured, interdependent information. SQL offers a structured access to the data, while XML still requires efficient handling of data streams where sometimes only small parts of the stream are relevant for the current purpose. If native XML databases should gain attention in the future, then the interface can be adjusted accordingly. For transferring meta data from one database to another, MiningMart offers an XML-based import/export

facility, already. Together with the flexibility in the data model due to the two-level approach (see above), this feature allows the adaptation of cases to different environments.

4 Conclusions

Finding the right representation of data is one of the most crucial issues in the KDD process. To this end the MiningMart system offers support for modelling conceptual knowledge about the domain. The description of available data in a higher level representation language is the first step towards an understandable and operational case study in this framework. Relevant concepts of a domain are structured by a domain ontology, allowing to structure concepts by means of inheritance and to represent different kinds of relationships between concepts. This kind of domain model allows to structure the relevant concepts better than the facilities available in relational databases, and allows for a convenient handling of views in different contexts. With the MiningMart meta model M4 it is possible to set up operational sequences of preprocessing steps, making use of the conceptual descriptions of the data, only. This results in an increase of the interpretability and reusability of KDD processes.

Acknowledgement

The MiningMart project was funded by the European Union under the contract number IST-1999-11993.

References

1. Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. Crisp–dm 1.0. Technical report, The CRISP–DM Consortium, August 2000.
2. Jörg-Uwe Kietz, Anca Vaduva, and Regina Zücker. MiningMart: Metadata-driven preprocessing. In *Proceedings of the ECML/PKDD Workshop on Database Support for KDD*, September 2001.
3. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems*, page 14. IEEE Computer Society, 1996.
4. Katharina Morik and Martin Scholz. The MiningMart Approach to Knowledge Discovery in Databases. In Ning Zhong and Jiming Liu, editors, *Intelligent Technologies for Information Analysis*. Springer, 2003. to appear.
5. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350, 2001.